Xian-he Sun Wenyu Qu Ivan Stojmenovic
Wanlei Zhou Zhiyang Li Hua Guo
Geyong Min Tingting Yang Yulei Wu
Lei Liu (Eds.)

LNCS 8630

Algorithms and Architectures for Parallel Processing

14th International Conference, ICA3PP 2014
Dalian, China, August 24–27, 2014
Proceedings, Part I

1 Part I

Springer

Xian-he Sun   Wenyu Qu   Ivan Stojmenovic
Wanlei Zhou   Zhiyang Li   Hua Guo
Geyong Min   Tingting Yang   Yulei Wu
Lei Liu (Eds.)

Algorithms and Architectures for Parallel Processing

14th International Conference, ICA3PP 2014
Dalian, China, August 24-27, 2014
Proceedings, Part I

Springer

Volume Editors

Xian-he Sun
Illinois Institute of Technology, Chicago, IL, USA, e-mail: sun@iit.edu

Wenyu Qu
Dalian Maritime University, China, e-mail: wenyu@dlmu.edu.cn

Ivan Stojmenovic
University of Ottawa, ON, Canada, e-mail: ivan@site.ottawa.ca

Wanlei Zhou
Deakin University, Burwood, VIC, Australia, e-mail: wanlei.zhou@deakin.edu.au

Zhiyang Li
Dalian Maritime University, China, e-mail: lizy0205@gmail.com

Hua Guo
BeiHang University, Beijing, China, e-mail: hguo@buaa.edu.cn

Geyong Min
University of Bradford, UK, e-mail: g.min@brad.ac.uk

Tingting Yang
Dalian Maritime University, China, e-mail: yangtingting820523@163.com

Yulei Wu
Chinese Academy of Sciences, Beijing, China, e-mail: yulei.frank.wu@gmail.com

Lei Liu
Shandong University, Jinan City, China, e-mail: l.liu@sdu.edu.cn

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-11196-4 e-ISBN 978-3-319-11197-1
DOI 10.1007/978-3-319-11197-1
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014947719

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper
Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Welcome to the proceedings of the 14th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2014) held in Dalian, China.

ICA3PP 2014 is the 14th in this series of conferences started in 1995 that are devoted to algorithms and architectures for parallel processing. As applications of computing systems have permeated in every aspect of daily life, the power of computing system has become increasingly critical. This conference provides a forum for academics and practitioners from countries around the world to exchange ideas for improving the efficiency, performance, reliability, security, and interoperability of computing systems and applications.

It is our great honor to introduce the program for the conference. Thanks to the Program Committee's hard work, we were able to finalize the technical program. In the selection process, each paper was assigned to at least 4 PC members as reviewers. The authors and those PC members from the same institution were separated in the reviewing process to avoid conflicts of interests. We received 285 submissions from all over the world. The large number of submissions indicated continued excitement in the field worldwide. The manuscripts have been ranked according to their original contribution, quality, presentation, and relevance to the themes of the conference. In the end, 70 (24.56%) papers were accepted as the main conference papers and inclusion in the conference.

ICA3PP 2014 obtained the support of many people and organizations as well as the general chairs whose main responsibility was various tasks carried out by other willing and talented volunteers. We want to express our appreciation to Professor Xian-He Sun for accepting our invitation to be the keynote/invited speaker.

We would like to give our special thanks to the program chairs of the conference for their hard and excellent work on organizing the Program Committee, outstanding review process to select high-quality papers, and making an excellent conference program. We are grateful to all workshop organizers for their professional expertise and excellence in organizing the attractive workshops/symposia, and other committee chairs, advisory members and PC members for their great support. We appreciate all authors who submitted their high-quality papers to the main conference and workshops/symposia.

We thank all of you for participating in this year's ICA3PP 2014 conference, and hope you find this conference stimulating and interesting.

July 2014

Ivan Stojmenovic
Wanlei Zhou

Organization

General Chairs

Ivan Stojmenovic	Ottawa University, Canada
Wanlei Zhou	Deakin University, Australia

Program Chairs

Xianhe Sun	Illinois Institute of Technology, USA
Wenyu Qu	Dalian Maritime University, China

Publicity Chairs

Jaime Lloret Mauri	Polytechnic University of Valencia, Spain
Al-Sakib Khan Pathan	International Islamic University Malaysia, Malaysia

Publication Chair

Yang Xiang	Deakin University, Australia

Steering Committee Chairs

Andrzej Goscinski	Deakin University, Australia
Yi Pan	Georgia State University, USA
Yang Xiang	Deakin University, Australia

Workshop Chairs

Mianxiong Dong	National Institute of Information and Communications Technology, Japan
Lei Liu	Shandong University, China

Local Organizing Chair

Zhiyang Li	Dalian Maritime University, China

Registration Chair

Weijiang Liu	Dalian Maritime University, China

Finance Chair

Zhaobin Liu	Dalian Maritime University, China

Web Chairs

Yang Shang	Dalian Maritime University, China
Tingting Wang	Dalian Maritime University, China

Program Committee Members

Zafeirios Papazachos	Queen's University of Belfast, UK
Paolo Trunfio	University of Calabria, Italy
Chao-Tung Yang	Tunghai University, Taiwan
Yong Zhao	University of Electronic Science and Technology of China, China
Xingquan (Hill) Zhu	Florida Atlantic University, USA
Giandomenico Spezzano	ICAR-CNR, Italy
Yasuhiko Takenaga	The University of Electro-Communications, Japan
Sushil Prasad	University of Georgia, USA
Tansel Ozyer	TOBB University of Economics and Technology, Turkey
Deng Pan	Florida International University, USA
Apostolos Papadopoulos	Aristotle University of Thessaloniki, Greece
Eric Pardede	La Trobe University, Australia
Karampelas Panagiotis	Hellenic American University, Greece
Paul Lu	University of Alberta, Canada
Kamesh Madduri	Penn State University, USA
Ching-Hsien Hsu	Chung Hua University, Taiwan
Muhammad Khurram Khan	King Saud University, Saudi Arabia
Morihiro Kuga	Kumamoto University, Japan
Weiwei Fang	Beijing Jiaotong University, China
Franco Frattolillo	Università del Sannio, Italy
Longxiang Gao	Deakin University, Australia
Javier García	University Carlos III, Spain
Michael Glass	University of Erlangen-Nuremberg, Germany
David E. Singh	Universidad Carlos III de Madrid, Spain
Marion Oswald	TU Wien, Austria
Rajkumar Buyya	The University of Melbourne, Australia

Yue-Shan Chang	National Taipei University, Taiwan
Christian Engelman	Oak Ridge National Lab, USA
Alessio Bechini	University of Pisa, Italy
Hideharu Amano	Keio University, Japan
Wei Wei	Xi'an University of Technology, China
Toshihiro Yamauchi	Okayama University, Japan
Bo Yang	University of Electronic Science and Technology of China, China
Laurence T. Yang	St. Francis Xavier University, Canada
Sherali Zeadally	University of the District of Columbia, USA
Sotirios G. Ziavras	NJIT, USA
Gennaro Della Vecchia	Gennaro Della Vecchia - ICAR-CNR, Italy
Olivier Terzo	Istituto Superiore Mario Boella, Italy
Hiroyuki Tomiyama	Ritsumeikan University, Japan
Tomoaki Tsumura	Nagoya Institute of Technology, Japan
Luis Javier García Villalba	Universidad Complutense de Madrid (UCM), Spain
Gaocai Wang	Guangxi University, China
Chen Wang	CSIRO ICT Centre, Australia
Martine Wedlake	IBM, USA
Wei Xue	Tsinghua University, China
Edwin Sha	University of Texas at Dallas, USA
Sachin Shetty	Tennessee State University, USA
Ching-Lung Su	National Yunlin University of Science and Technology, Taiwan
Anthony Sulistio	High Performance Computing Center Stuttgart (HLRS), Germany
Magdalena Szmajduch	Cracow University of Technology (CDN Partner Cracow), Poland
Jie Tao	University of Karlsruhe (Karlsruhe Institute of Technology), Germany
Dana Petcu	West University of Timisoara, Romania
Florin Pop	University Politehnica of Bucharest, Romania
Rajeev Raje	Indiana University-Purdue University Indianapolis, USA
Francoise Sailhan	CNAM, France
Subhash Saini	NASA, USA
Erich Schikuta	University of Vienna, Austria
Alba Amato	Second University of Naples, Italy
Cosimo Anglano	Università del Piemonte Orientale, Italy
Ladjel Bellatreche	ENSMA, France
Ateet Bhalla	Oriental Institute of Science and Technology, India

Surendra Byna	Lawrence Berkeley National Lab, USA
Aleksander Byrski	AGH University of Science and Technology, Poland
Juan M. Marin	University of Murcia, Spain
Francesco Moscato	Second University of Naples, Italy
Hirotaka Ono	Kyushu University, Japan
Fabrizio Petrini	IBM Research, USA
Stefano Marrone	Second University of Naples, Italy
Alejandro Masrur	Technology University of Munich, Germany
Susumu Matsumae	Saga University, Japan
Wei Lu	Keene University, USA
Amit Majumdar	San Diego Supercomputer Center, USA
Tomas Margalef	Universitat Autonoma de Barcelona, Spain
Che-Rung Lee	National Tsing Hua University, Taiwan
Keqin Li	State University of New York at New Paltz, USA
Mauro Iacono	Second University of Naples, Italy
Shadi Ibrahim	Inria, France
Helen Karatza	Aristotle University of Thessaloniki, Greece
Soo-Kyun Kim	PaiChai University, Korea
Edmund Lai	Massey University, New Zealand
Karl Fuerlinger	Ludwig-Maximilians-University Munich, Germany
Jose Daniel Garcia	University Carlos III of Madrid, Spain
Harald Gjermundrod	University of Nicosia, Cyprus
Houcine Hassan	Universidad Politecnica de Valencia, Spain
Raphaël Couturier	University of Franche-Comté, France
Eugen Dedu	University of Franche-Comté, France
Ciprian Dobre	University Politehnica of Bucharest, Romania
Massimo Cafaro	University of Salento, Italy
Ruay-Shiung Chang	National Dong Hwa University, Taiwan
Dan Chen	University of Geosciences, China
Zizhong (Jeffrey) Chen	University of California at Riverside, USA
Jing Chen	National Cheng Kung University, Taiwan
Carmela Comito	University of Calabria, Italy
Yujie Xu	Dalian Maritime University, China
Natalija Vlajic	York University, Canada
Kenji Saito	Keio University, Japan
Thomas Rauber	University of Bayreuth, Germany
Pilar Herero	Universidad Politecnica de Madrid, Spain
Tania Cerquitelli	Politecnico di Torino, Italy
Tzung-Shi Chen	National University of Tainan, Taiwan
David Expósito	University Carlos III, Spain
Peter Strazdins	The Australian National University, Australia
Uwe Tangen	Ruhr-Universitaet Bochum, Germany

Luca Tasquier	Second University of Naples, Italy
Rafael Santos	National Institute for Space Research, Brazil
George Bosilca	University of Tennessee, USA
Esmond Ng	Lawrence Berkeley National Lab, USA
Laurent Lefevre	Laurent Lefevre, Inria, University of Lyon, France
Giuseppina Cretella	Second University of Naples, Italy
Gregoire Danoy	University of Luxembourg, Luxembourg
Bernabe Dorronsoro	University of Lille 1, France
Massimo Ficco	Second University of Naples, Italy
Jorge Bernal Bernabe	University of Murcia, Spain

Keynote

C-AMAT: Concurrent Data Access Model for the Big Data Era

Xian-He Sun

Illinois Institute of Technology, Chicago, USA
sun@iit.edu

Abstract. Scalable data management for big data applications is a challenging task. It puts even more pressure on the lasting memory-wall problem, which makes data access the prominent performance bottleneck for high-end computing. High-end computing is known for its massively parallel architectures. A natural way to improve memory performance is to increase and utilize memory concurrency to a level commensurate with that of high-end computing. We argue that substantial memory concurrency exists at each layer of current memory systems, but it has not been fully utilized. In this talk we reevaluate memory systems and introduce the novel C-AMAT model for system design analysis of concurrent data accesses. C-AMAT is a paradigm shift to support sustained data accessing from a data-centric view. The power of C-AMAT is that it has opened new directions to reduce data access delay. In an ideal parallel memory system, the system will explicitly express and utilize parallel data accesses. This awareness is largely missing from current memory systems. We will review the concurrency available in modern memory systems, present the concept of C-AMAT, and discuss the considerations and possibility of optimizing parallel data access for big data applications. We will also present some of our recent results which quantize and utilize parallel I/O following the parallel memory concept.

Keywords: Big Data; Parallel memory system; Data access model

1 Bio-Short version

Dr. Xian-He Sun is a Distinguished Professor of Computer Science and the chairman of the Department of Computer Science at the Illinois Institute of Technology (IIT). He is the director of the Scalable Computing Software laboratory at IIT and a guest faculty in the Mathematics and Computer Science Division at the Argonne National Laboratory. Before joining IIT, he worked at DoE Ames National Laboratory, at ICASE, NASA Langley Research Center, at Louisiana State University, Baton Rouge, and was an ASEE fellow at Navy Research Laboratories. Dr. Sun is an IEEE fellow and is known for his memory-bounded speedup model, also called Sun-Nis Law, for scalable computing. His research interests include parallel and distributed processing, memory and I/O systems, software systems for big data applications, and performance evaluation. He has over 200 publications and 4 patents in these areas. He is a former IEEE CS distinguished speaker and former vice chair of the IEEE Technical Committee on Scalable Computing, and is serving and served on the editorial board of most of the leading professional journals in the field of parallel processing. More information about Dr. Sun can be found at his web site `www.cs.iit.edu/~sun/`.

Table of Contents – Part I

Table of Contents – Part II

Porting the Princeton Ocean Model to GPUs

Shizhen Xu[1,3], Xiaomeng Huang[1,2,3], Yan Zhang[1,2,3], Yong Hu[1,3], Haohuan Fu[1,2,3], and Guangwen Yang[1,2,3]

[1] Ministry of Education Key Laboratory for Earth System Modeling
[2] Center for Earth System Science, Tsinghua University, 100084
[3] Joint Center for Global Change Studies, Beijing, 100875, China
{hxm,haohuan,ygw}@tsinghua.edu.cn,
{yan-zhang12,huyong11,xsz12}@mails.tsinghua.edu.cn

Abstract. While GPU is becoming a compelling acceleration solution for a series of scientific applications, most existing work on climate models only achieved limited speedup. It is due to partial porting of the huge code and the memory bound inherence of these models. In this work, we design and implement a customized GPU-based acceleration of the Princeton Ocean Model (gpuPOM). Based on Nvidia's state-of-the-art GPU architectures (K20X and K40m), we rewrite the original model from the Fortran into the CUDA-C completely. Several accelerating methods, including optimizing memory access in a single GPU, overlapping communication and boundary operations among multiple GPUs, are presented. The experimental results show that the gpuPOM on one K40m GPU achieves 6.9-fold to 17.8-fold speedup and 5.8-fold to 15.5-fold speedup on one K20X GPU comparing with different Intel CPUs. Further experiments on multiple GPUs indicate that the performance of the gpuPOM on a super-workstation containing 4 GPUs is equivalent to a powerful cluster consisting of 34 pure CPU nodes with over 400 CPU cores.

1 Introduction

There is no doubt that high-resolution climate modeling is crucial for simulating global and regional climate change. Most research groups in climate modeling have established their own roadmaps for high-resolution ranging from several kilometers down to hundreds of meters. The need for high-resolution exposes serious problems because the time consumed in running high-resolution climate models remains a significant scientific and engineering challenge.

Recent years, many scientific codes have been ported to the GPU and well suited to the GPU. In the area of climate models, most of the previous work achieved different levels of speedup for entire models on GPUs. For instance, Michalakes et al accelerated a computationally intensive microphysics process of the Weather Research and Forecast (WRF) model with a speedup of nearly 25x that speedups the entire WRF model by only 1.23x[1]; Shimokawabe et al fully accelerated the ASUCA model – a production and non-hydrostatic weather model – on 528 Nvidia Tesla GT200 GPUs and achieves 15TFlops[2]; [3] accelerated a full huge operational weather forecasting model COSMO and achieved 2.8X speedups for its dynamic core.

According to our analysis, further speedup of climate models is limited by two reasons. Firstly, the partial GPU porting limits the performance of the whole application. Many scientific models suffer from a flat performance profile during GPU accerlation. In CAM[4], the most single expensive subroutine only accounts for about 10% of total runtime and most subroutines accounts for less than 5%, which is the same in mpiPOM. According to the Amdahl law, the whole model can not be significantly speedup such as the GPU accerlation of CAM [4], ROMS [5], WRF [1] and HOMME [6], but it is sometimes a compromised approach because of the huge code. Secondly, climate models is mainly bounded by memory access, especially for their dynamic cores[3]. Great work has been done in the full GPU acceleration of COSMO [3], ASUCA [2] and NIM [7], including all the dynamic cores and some portion of physics. But the memory-bound problem exists and can be further eased through better use of state-of-the-art GPU memory hierarchy.

The objective of our study is to shorten the computation time of high-resolution ocean models by parallelizing their existing model structures using the GPU. With the representative ocean model, the mpiPOM, used as an example, we demonstrate how to parallelize an ocean model to make it effectively run on GPU architecture. Using state-of-the-art GPU architecture, we first rewrite the entire mpiPOM model from the original Fortran version into a new CUDA-C version. We call the new version gpuPOM. Then, we design and implement several different optimizing methods from two levels, such as computation optimization in a single GPU, communication optimization among multiple GPUs.

In terms of computation, we concentrate on memory access optimization and making better use of GPU's memory hierarchy. We deploy a four-categories optimizations, including using read-only data cache, local memory blocking, loop fusion and subroutine fusion, that are especially effective for climate models. The experimental results demonstrate that one K40m GPU achieves 6.9-fold to 17.8-fold speedup and one K20X achieves 5.8-fold to 15.5-fold speedup over different Intel multi-core CPUs.

In terms of communication, we concentrate on the fine-grained overlapping between the inner-region computation and the outer-region communication and updating. With the new design, multiple GPUs in one node can communicate directly bypassing the CPU. In addition, with the fine-grained control of the CUDA streams and their priorities, inner-region computation can execute concurrently with outer-region communication and updating.

To understand the accuracy, performance and scalability of the gpuPOM, we build a customized super-workstation with four K20X GPUs inside. Experimental results show that the performance of the gpuPOM running on this super-workstation can compare with a powerful cluster consisted of 34 pure CPU nodes with over 400 CPU cores, which means this novel gpuPOM version provides a fast and attractive solution for ocean scientists to conduct simulation research.

The remainder of this paper is organized as follows. We review the mpiPOM in Sec. 2, and introduce the structure of the gpuPOM in Sec. 3. In Sec. 4, we introduce four optimizations to efficiently use GPU's memory hierarchy for the gpuPOM. In Section 5, we present detailed techniques about communication optimization among multiple GPUs. We provide the corresponding experimental results about correctness, performance and scalability in Section 6 and conclude our work in Section 7.

2 The mpiPOM

The mpiPOM[8] is a widely used parallel branch of POM based on Message Passing Interface (MPI) and retains part the physics package of the original POM[9]. POM is a powerful regional ocean model and has been used in a wide-range of applications such as circulation and mixing processes in rivers, seas and oceans.

The mpiPOM solves the primitive equation under hydrostatic and boussinesq approximations. In the horizontal, spatial derivatives are computed using centered-space differencing on a staggered Arakawa C-grid. In the vertical, the mpiPOM supports terrain-following sigma coordinates and a fourth-order scheme option to reduce the internal pressure-gradient errors. The mpiPOM uses the classical time-splitting technique to separate the vertically integrated equations (external mode) from the vertical structure equations (internal mode). The external mode calculation is responsible for updating surface elevation and the vertically averaged velocities. The internal mode calculation results in updates for velocity, temperature and salinity, in addition to the turbulence quantities. The three-dimensional internal mode and the two-dimensional external mode are both integrated explicitly using a second-order leapfrog scheme. These two modules are the most time consuming modules of mpiPOM.

To demonstrate the memory-bound problem, the Performance API(PAPI)[10] is used to estimate floating point operations count and memory access(store/load) instructions count. Results shows that the computational intensity(flops/byte) of the mpiPOM is about 1:3.3, while that provided by SandyBridge E5-2670 CPUs is about 7.5:1. Moreover, data are mostly streamed from memory and shows little locality, which means that mpiPOM is mainly bounded by memory access. The wider memory bandwidth of GPU has attracted us a lot and to the best of our knowledge, our work is the first GPU porting of the POM.

3 Structure of the gpuPOM

The flowchart of the gpuPOM is illustrated in Fig. 1. The main difference between mpiPOM and gpuPOM is that CPU in the gpuPOM implementation is only responsible for the initializing and output. The gpuPOM begins with initializing the relevant arrays on CPU host and then copy these data to GPU. GPU will deal with all the computation including external mode and internal mode, and so on. During the computation, the variables required for output like

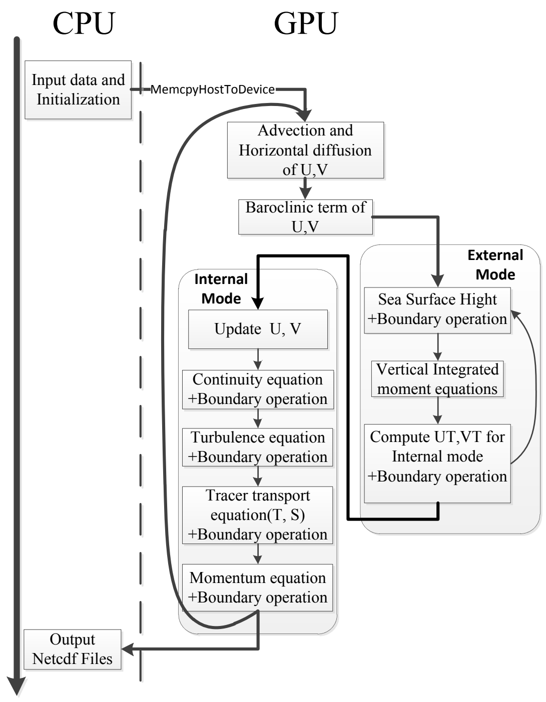

CPU
GPU
Input data and Initialization
MemcpyHostToDevice
Advection and Horizontal diffusion of U,V
Baroclinic term of U,V
External Mode
Sea Surface Hight +Boundary operation
Vertical Integrated moment equations
Compute UT,VT for Internal mode +Boundary operation
Internal Mode
Update U, V
Continuity equation +Boundary operation
Turbulence equation +Boundary operation
Tracer transport equation(T, S) +Boundary operation
Momentum equation +Boundary operation
Output Netcdf Files

Fig. 1. Flowchart of the gpuPOM

velocity and surface elevation will be copied back to CPU host and then written to disk at a constant frequency.

In Fig. 1, each module represents a part of the primitive equations and is implemented by several subroutines. We retained the overall structure of the original code and rewrote all the subroutines with about 70 CUDA kernel functions. Thus data always resides on GPU during the computation.

In our implementation , the 3D arrays of variables are stored sequentially in the order of x, y, z (i, j, k ordering) and the 2D arrays are in the order of x, y, which is the same with the original code. Each GPU thread specifies a (x, y) point in horizontal direction and performs all the calculations from surface to bottom(30 to 50 points). The threadblocks are configured with $(32, 4, 1)$ threads and similar performance can be achieved if configured with $(32, 8, 1)$ or $(32, 16, 1)$ for the K20X GPU we use.

4 Optimizations on Single GPU

Here, we propose four key optimizations to fully utilize the memory hierarchy of GPU and describe how they apply to the algorithms in the gpuPOM.

(A) Using Read-Only Data Cache. Nvidia's Kepler architecture GPU(sm3.5) introduces a new 48KB directly-access read-only data cache which is known as texture cache in previous generation. Read-only data cache adds another way to cache global memory access and we use it to cache data that are read among adjacent threads besides the well-known shared memory. A decoration of "const __restrict__" qualifiers to the parameter pointers will explicitly direct the compiler to do this optimization. After adding such a decoration, the "LDG.E" instruction will appear in the disassembling code and Nvidia's Visual Profiler(NVVP) will show the read-only data cache is actually utilized.

Taking the calculation of advection and horizontal diffusion term as example. Since the mpiPOM adopts Arakawa C-grid in horizontal direction, the update of temperature at time n+1 $Tf(i,j,k)$ needs values of $u(i,j,k)$ and $v(i,j,k)$. In addition, it also needs the three values of aam, Tb, T on point (i,j,k), $(i,j-1,k)$, and $(i-1,j,k)$. It means that the horizontal kinematic viscosity array aam, the temperature at time n-1 Tb and the temperature at time n T have to be accessed by 3 times(3-points stencil) in one step. The use of the read-only data cache improves performance of this part by 18.8%.

(B) Local Memory Blocking. Cache blocking is an optimization technique to explicit load a small subset of much larger dataset into on-chip faster memory. And repeatedly accessing this small subset can save a lot of memory bandwidth. For K20X GPU, global memory access can not be cached in L1 cache and it is reserved for local memory access such as stack data declared in kernel functions[11]. So we name this optimization local memory blocking.

For the subroutines about vertical diffusion and source/sink terms, chasing method is used to solve a tridiagonal matrix of depth-direction for each grid point individually. In the original mpiPOM, the 3D temporary arrays like ee, gg, which store row transformation coefficients, are streamed from memory. In our implementation, each thread performs a column calculation from surface to bottom. We declare 1D arrays ee_new, gg_new in local memory to replace the original 3D global arrays. Their size equals to the vertical levels of ocean, kb, which is usually a very small value (30 to 50).

In the chasing method, these local arrays are accessed twice within one thread, once from k=0 to k=kb-1 and once from k=nz-1 to k=0. After block the depth-direction arrays in local memory, L1 cache is fully utilized although some of them have to be spilled to global memory. The use of the local memory blocking improves performance of the vertical diffusion part by 35.3%.

(C) Loop Fusion. Loop fusion is an optimization to cache several scalar variables in registers by fusing several loops into one. For example, if u(i,j,k) is read several times in different loops, we can fuse these loops into one. Thus u(i, j, k) is read from global memory in the first time, and then repeatedly read from a register. And it can apply to almost all subroutines in mpiPOM.

Moreover, a computation pattern exists in many subroutines of mpiPOM that local temporary arrays are calculated in one i,j,k-loop and read in the next i,j,k-loop. Although it brings clean code, it brings extra burden for memory bandwidth as well. After loop fusion, we can eliminate these local temporary arrays, and great speedup comes at the cost of more complex code in one loop.

But, it is worth mentioning that loop fusion can not be applied to every case, and local memory blocking is such one. If an array has to be traversed from k=0 to k=kb-1 first, and then from k=kb-1 to k=0, we can not fuse them into one.

Take the most time consuming kernel $profq\_1$ as an example. After Loop fusion, the device memory transactions decrease by 57%, while the registers used per thread increase from 46 to 72, as reported in NVVP. Although the occupancy achieved decreases from 61.1% to 42.7%, the performance of this kernel is improved by 28.6%.

(D) Subroutine Fusion. Similar to loop fusion, we can also fuse subroutine in which similar formulas are calculated and same arrays are accessed. Subroutine fusion here refer to fusing two loops in two subroutines. For example, the *advv* and *advu* subroutines of the mpiPOM calculate advection in longitude and latitude respectively and they can be fused into one subroutine. Also, there exist subroutines that are called several times to calculate different tracers. For example, the $proft$ subroutine is called twice for temperature and salinity respectively, and they can be fused into one to calculate these two tracers simultaneously.

This optimizations benefits because it turns several global memory accesses into one, which is the same with loop fusion. We have fused subroutines *advu*/ *advv*, $profu/profv$ into new subroutines *advuv*, $profuv$. And we expand the content of *advq*, $proft$ to calculate several tracers simultaneously. The use of the subroutine fusion improves performance of these subroutines by 28.8%.

5 Communication Optimizations among Multiple GPUs

In this section, we present the optimizing strategies used to harness the computing power of multiple GPUs within one node. To utilize multiple GPUs, an effective domain decomposition method and communication method should be employed. We split the domain along both the x and y directions (2-D decomposition) and assign each MPI process for one subdomain, following [12].

State-of-the-art MPI libraries, such as OpenMPI and MVAPICH2, have announced that MPI communication directly from GPU memory, which is known as CUDA-aware MPI, is supported. We first try MVAPICH2 to implement direct communication among multiple GPUs. However, we found that the boundary operation and MPI communication occupied more than 10% of the total runtime after GPU porting. Previous solutions [2][13] on large amount of GPU clusters focus on the explicit involvement of CPU in communication. We design a novel communication method to make GPUs directly communicate with each other to reduce latency.

To fully overlap the boundary operations and MPI communications with computation, we adopt the data decomposition method shown in Fig. 2. The data

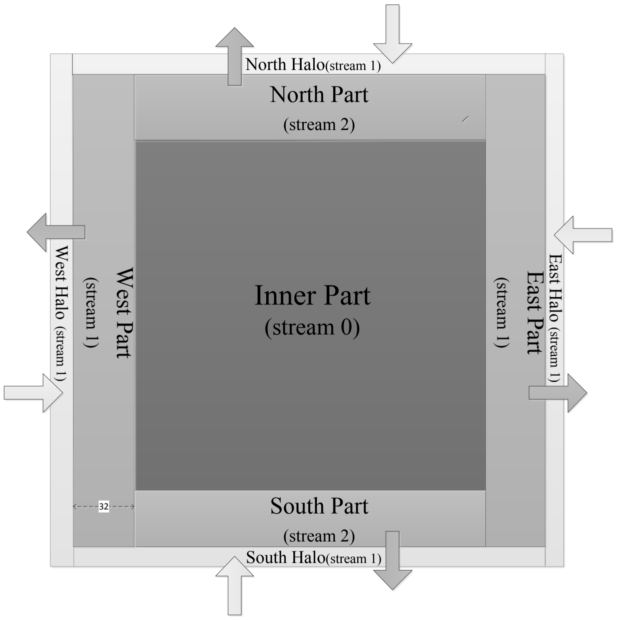


Fig. 2. Data decomposition in the gpuPOM

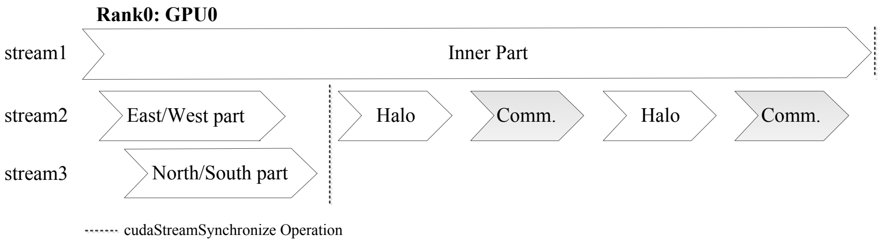


Fig. 3. The workflow of multiple streams on the GPU. The "inner/east/west/north/south part" and "Halo" refer to computation and update of corresponding part. "Comm." refers to communication between processes, which implies synchronization.

region is decomposed into three parts: the inner part, the outer part, and the halo part. The outer part includes east/west/north/south part, and the halo part also includes east/west/north/south halos to exchange data with neighbors. In CUDA, a stream is a sequence of commands that execute in order; in addition, different streams can execute concurrently with different priorities. In our design, the inner part, which is the most time-consuming part with the largest workload that is used to overlap other parts, is allocated stream 1 in which to execute. The east/west outer part is allocated to stream 2 and the north/south outer part is allocated to stream 3. In the east/west outer part, the width is set to 32 to ensure coalesced memory access in a warp to improve performance. The halo part is also allocated to stream 2.

The workflow of multiple streams on the GPU is shown in Fig. 3. The outer parts are normal kernel functions that can run in parallel with the inner part through different streams. The communication operations are implemented by *cudaMemcpyAsync*, which will be detailed later. The corresponding synchronization operation between the CPU and the GPU or among MPI processes are implemented with *cudaStreamSynchronize* function and *MPI_barrier* function. To hide the subsequent communication and boundary operations by the inner part, stream 2 and stream 3 for the outer part have higher priority to preempt the computing resource from stream 1 at any time.

Current CUDA-aware MPI implementation such as MVAPICH2 is not suitable for the "Comm." part in Fig. 3. We found the two-sided MPI functions *MPI_Send* and *MPI_Recv* will block the current stream such that the concurrency pipeline is broken. The probable cause is synchronous *cudaMemcpy* function is called in the current implementation of *MPI_Send* and *MPI_Recv*, according to [14]. Moreover, the implementation of non-contiguous MPI datatype for communication between GPUs is not efficient enough for the gpuPOM. The computation time of many kernels is about a few hundred microseconds to a few milliseconds while the MPI latency for our message size is about the same, which means the outer part update and communication can not fully be overlapped.

From CUDA 4.1, the Inter-Process Communication (IPC) feature has been introduced to facilitate direct data copy among multiple GPU buffers that are allocated by different processes. The IPC is implemented by creating and exchanging memory handles among processes and obtaining the device buffer pointers of others. This feature has been utilized in CUDA-aware MPI libraries to optimize communications within a node. Therefore, we decided to implement the communication among multiple GPUs by calling the low-level IPC functions and asynchronous CUDA memory copy functions directly, instead of using high-level CUDA-aware MPI functions. Our communication optimizations among multiple GPUs are mainly implemented in the following two features.

First, we put the phases of creating, exchanging and opening relevant memory handles into the initialization phase of the gpuPOM, which is executed only once. This method can remove the overhead of IPC memory handle operations during each MPI communication operation. The *cudaMemcpyAsync* and *cudaMemcpy2DAsync* functions with the corresponding device buffer pointers of neighbor processes replaces the original MPI functions.

Second, we take full consideration of the architecture of our platform in which 4 GPUs are connected with two different I/O Hubs (IOHs). As illustrated in Fig. 4, there are two Intel SandyBridge CPUs that connect two GPUs. Both the CPUs are themselves connected through Intel QuickPath Interconnect (QPI). Notation ① means that the communications between GPUs are connected with the same IOH support Peer-to-Peer (P2P) access. Notation ② represents the

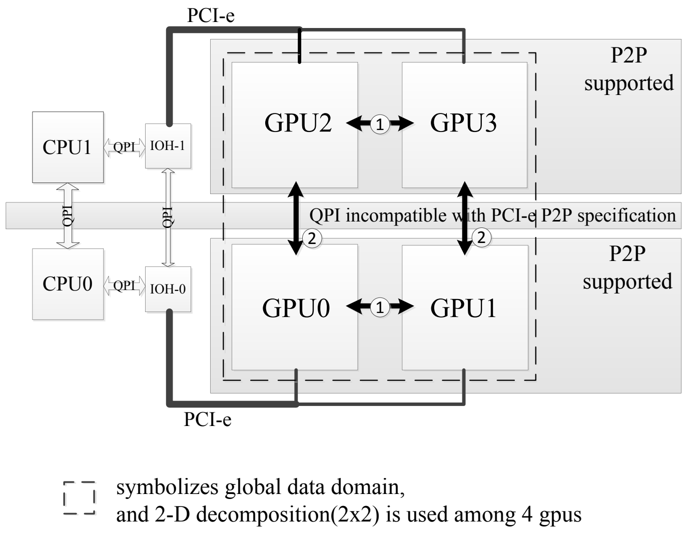

PCI-e
P2P supported
GPU2
GPU3
1
CPU1
QPI
IOH-1
QPI
QPI
QPI incompatible with PCI-e P2P specification
2
2
P2P supported
CPU0
QPI
IOH-0
GPU0
GPU1
1
PCI-e
symbolizes global data domain,
and 2-D decomposition(2x2) is used among 4 gpus

Fig. 4. Communications pattern among multiple GPUs in one node

communications in which P2P access is not supported. During the initialization phase, MPI_Rank 0 (context on GPU-0) must switch its context to GPU-2 temporally and open corresponding memory handles to obtain the device buffer pointers of rank 2, in order to communicate with MPI_Rank 2 (context on GPU-2). For communications between GPUs on the same IOH, context switching is not necessary. Although the functions *cudaMemcpyAsync* and *cudaMemcpy2DAsync* are used for data transfer in the communication of both ① and ②, the NVVP shows that ① does a device-to-device memory copy that bypasses the CPU, whereas ② does a device-to-host and a host-to-device memory copy that implies the involvement of CPU. 2-D decomposition in Fig. 4 is used as an example to demonstrate our design can easily extend to 8 or more GPUs within one node.

6 Experiments

In this section, we first describe the specification of our platform and compare methods used to validate the correctness of the gpuPOM. Furthermore, we present the performance and scalability of the gpuPOM on the GPU platform in comparison with the mpiPOM on the CPU platform.

6.1 Platform Setup

The GPU platform used in our single GPU experiments is a workstation consisting of one K20X GPU and one K40m GPU, while the platform used in scaling experiments is the same workstation consisting of four K20X GPUs. The CPUs are 8-core Intel E5-2670. Every two GPUs are connected with one IO Hub. The programs on this platform are test with Intel compiler v14.0.2, Intel MPI Library v4.1.3 and CUDA 6.0 Toolkit.

For the purposes of comparison, the CPU platform used in our experiments is a cluster consisting of Intel X5670, E5-2670 and E5-2670v2 CPUs and connected with Infiniband QDR. The original mpiPOM code is benchmarked with its initial compiler flags(i.e., -O3 -precise) using the Intel compiler and MPI library. We also use MVAPICH2 v1.9 to test the communication effects among multiple GPUS, and compare the results with our method.

6.2 The Test Case and the Verification of Accuracy

We adopt one "dam break" test case(single precision) to verify the correctness and test the performance and the scalability of the gpuPOM. It simulates dam break (warm_South & cold_North) in f-plane zonal channel. The domain size of this test case is 962×722 horizontal grid points and 51 vertical sigma levels. The metrics of seconds per simulation day, which is the walltime it requires to run 24 hours in the simulation, is measured to compare the performance.

To verify accuracy, we check the binary output files output from the original mpiPOM and the gpuPOM. This testing method is also used in the GPU-porting of ROMs [5]. As introduced in [15], the same inputs will give identical results for individual IEEE-754 operations except in a few special cases. These cases can be classified into three categories: different operations orders, different instruction sets and different implementations of math libraries. For the first in our study, the parallelization of the mpiPOM does not change the order of each floating point operation and we benefit from this. For the second case in our study, the GPUs own fused multiply-add (FMA) instruction while the CPU does not in our CPU platform. Because this instruction might cause a difference in the numerical results, we disable FMA instructions with the "-fmad=false" compiler flag for the gpuPOM. For the third case in our study, the value of exponent used in the GPU has a maximum of 2 ulp errors [11]. Fortunately, in the execution path of test cases described above, the power of the exponent functions remains unchanged over the entire simulation. Therefore, we accomplish this function on the CPU during the initialization phase and copy the results to the GPU for later reuse. The experimental results demonstrate that the output variables regarding velocity, temperature, salinity and sea surface height are identical, which can validate our GPU implementation.

6.3 Single GPU Performance

The original mpiPOM is tested on three conventional Intel CPU platforms. For K20X GPU, we discard the test case of the domain size with 1922*722*51 grids, for the data in this case has exceeded the on-board memory of K20X(6GB). Execution time on 8 IvyBridge CPUs in the table is also left blank because we only have access to 6 IvyBridge CPUs.

Table 1. Performance comparison between single GPU and CPUs

		Numbers of CPUs/Cores					
Platform	Grid	1-core	1-CPU	4-CPUs	8-CPUs	12-CPUs	16-CPUs
Intel	722*482*51	2276.2	**710.2**	165.5	97.9	64.3	47.8
Westmere	962*722*51	4795.0	**1541.8**	343.3	198.3	131.5	98.9
X5670	1922*722*51	9884.5	**3236.9**	738.0	410.1	265.9	199.5

Platform	Grid	1-core	1-CPU	2-CPUs	4-CPUs	6-CPUs	8-CPUs
Intel	722*482*51	1718.5	**363.6**	170.4	85.2	57.8	43.8
SNB	962*722*51	3388.8	**796.6**	350.1	169.0	112.7	86.5
E5-2670	1922*722*51	6964.3	**1720.0**	793.6	361.7	245.0	174.4
Intel	722*482*51	1770.9	**266.5**	128.6	67.0	44.5	
IVB	962*722*51	3011.2	**581.1**	261.0	127.9	86.7	
E5-2670v2	1922*722*51	6257.5	**1257.7**	574.4	272.1	178.3	

		Optimizations					
			read-only	local	loop	subroutine	ECC-off
Platform	Grid	origin	data cache	memory	fusion	fusion	Boost
Nvidia	722*482*51	98.2	90.0	81.2	58.9	58.0	**50.1**
K20X	962*722*51	199.1	183.0	163.1	117.5	115.6	**99.7**
	722*482*51	96.9	94.4	82.6	58.9	58.2	**45.7**
Nvidia	962*722*51	201.8	196.7	169.2	119.1	117.9	**92.0**
K40	1922*722*51	417.7	403.7	336.4	237.0	235.4	**181.5**

It is obvious in the table that our gpuPOM on one K40 GPU has achieved about 17.8-fold, 9.5-fold, 6.9-fold speedup compared with Intel Westmere, Sandy-Bridge, IvyBridge CPUs respectively. The speedups on K20X GPU are 15.5-fold, 8-fold, and 5.8-fold. We further accelerate the gpuPOM by about 2 times with four optimizations and hardware tuning. The local memory blocking benefits a lot because the local memory is very suitable for the computation pattern of vertical diffusion part, where each thread calculates from bottom to surface and does not need to share values with adjacent threads. The loop fusion together with subroutine fusion contributes most in these optimizations, because a "fatter loop" can turn subsequent global memory accesses into registers accesses. This strategy can also be applied many existing scientific codes.

Further test with Performance API(PAPI) shows the performance of the gpuPOM on single K20X is 107.3Gflops in single precision for the 962*722*51 grid size. The low performance in Gflops reflects the memory-bounded problem in climate models. Previous work such as time skewing[16] can make a stencil computation compute bound by making use of data locality between different time-steps. But for real-world climate models including mpiPOM, the code is usually tens to hundreds of thousand lines and analyzing the dependency manually is tough. An automate tool to further optimize the mpiPOM and the gpuPOM is a part of future work.

6.4 Scalability

In the weak-scaling experiment, we test our communication overlapping design used in the gpuPOM and compare it with another two communication method as shown in Fig. 5. The "CUDA-aware MPI" refers to directly calling the MVAPICH2 library to fulfill MPI communication between buffers on two GPUs without overlapping. The "D2H2D MPI overlapping" refers to explicitly copying data from GPU to CPU and then doing conventional MPI communications, which is one of the key optimizations in [2]. The only difference between our design and "D2H2D MPI overlapping" is the implementation of "Comm." in Figure. 3. To maximize performance, the grid size for each GPU is set to 962 $\times$ 722 $\times$ 51. When using 4 GPUs with the implementation of "CUDA-aware MPI", more than 10% of the total runtime is consumed in executing the communication and boundary operations. This overhead does not exist in our communication overlapping method. Fig. 5 shows that it spends almost the same time when using different GPUs because the communication and boundary operations is almost fully overlapped with the inner part computation. To compare the gpuPOM on 4 GPUs with conventional CPUs, original mpiPOM is also benchmarked on 68 Intel X5670 CPUs with 408 cores. The time is 102.5s, 20% of which is cost in communication. It means the gpuPOM can compete with a cluster of pure CPU nodes with over 400 cores. For the current simulations, gpuPOM on a super-workstation provides a much faster and cost-effective way. For finer grid with much bigger data that may be needed in the future, gpuPOM has to be extended to a GPU cluster. It is a part of the future work.

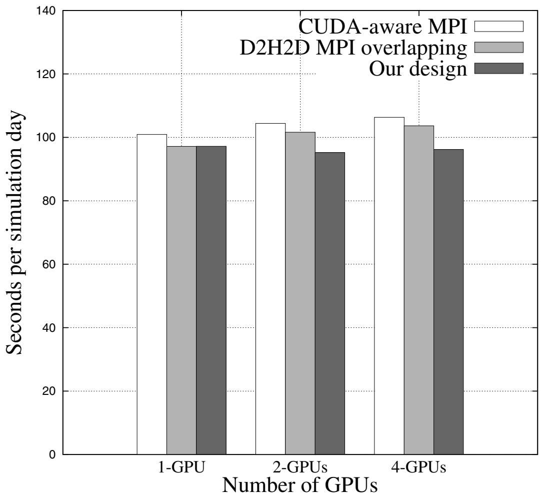


Fig. 5. The weak scaling test between our communication overlapping design and another two communication methods

Table 2. The strong scaling result of the gpuPOM

Number of GPUs	1-GPU	2-GPUs	4-GPUs
Time(s)	97.15	48.71	26.33
Efficiency	1.000	0.99	0.92

In the strong scaling experiment, we test the efficiency of the gpuPOM on multiple GPUs. Table 2 shows the strong scaling result of the gpuPOM on multiple GPUs. We fix the global grid size at $962 \times 722 \times 51$ and increase the amount of GPUs gradually. The results show that the strong scaling efficiency is 99% on 2 GPUs and 92% on 4 GPUs. A smaller subdomain will decrease the performance of the gpuPOM, this is because communication time can easily exceed the inner part computation time and cannot be overlapped.

7 Conclusion

In this paper, we note the memory bound problems in POM and provide a full GPU accelerated solution. Unlike partial GPU porting, such as WRF and ROMs, the gpuPOM does all the computation on the GPU. The main contribution of our work includes a better use of state-of-the-art GPU architecture, particularly regarding the memory subsystem and a new design of a communication and boundary operations overlapping approach. The gpuPOM on a super-workstation with 4 GPUs achieves over 400x speedup compared with original mpiPOM. For the users of the POM, this work can provide them a cost-effective and energy-effective way to run their ocean modeling.

Acknowledgment. This work is supported in part by a grant from the Natural Science Foundation of China(41375102), the National Grand Fundamental Research 973 Program of China (No. 2014CB347800) and the National High Technology Development Program of China (2011AA01A203).

References

1. Michalakes, J., Vachharajani, M.: Gpu acceleration of numerical weather prediction. Parallel Processing Letters 18(04), 531–548 (2008)
2. Shimokawabe, T., Aoki, T., Muroi, C., Ishida, J., Kawano, K., Endo, T., Nukada, A., Maruyama, N., Matsuoka, S.: An 80-fold speedup, 15.0 tflops full gpu acceleration of non-hydrostatic weather model asuca production code. In: IEEE 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1–11 (2010)
3. Fuhrer, O., Osuna, C., Lapillonne, X., Gysi, T., Bianco, M., Schulthess, T.: Towards gpu-accelerated operational weather forecasting. In: The GPU Technology Conference, GTC 2013 (2013)
4. Kelly, R.: Gpu computing for atmospheric modeling. Computing in Science & Engineering 12(4), 26–33 (2010)

5. Mak, J., Choboter, P., Lupo, C.: Numerical ocean modeling and simulation with cuda. In: IEEE OCEANS, pp. 1–6 (2011)
6. Carpenter, I., Archibald, R., Evans, K.J., Larkin, J., Micikevicius, P., Norman, M., Rosinski, J., Schwarzmeier, J., Taylor, M.A.: Progress towards accelerating homme on hybrid multi-core systems. International Journal of High Performance Computing Applications 27(3), 335–347 (2013)
7. Govett, M., Middlecoff, J., Henderson, T.: Running the nim next-generation weather model on gpus. In: IEEE, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 792–796 (2010)
8. Oey, L.Y., Lee, H.C., Schmitz, W.J.: Effects of winds and caribbean eddies on the frequency of loop current eddy shedding: A numerical model study. Journal of Geophysical Research: Oceans (1978–2012) 108(C10) (2003)
9. Blumberg, A.F., Mellor, G.L.: A description of a three-dimensional coastal ocean circulation model. Coastal and Estuarine Sciences 4, 1–16 (1987)
10. Browne, S., Dongarra, J., Garner, N., Ho, G., Mucci, P.: A portable programming interface for performance evaluation on modern processors. International Journal of High Performance Computing Applications 14(3), 189–204 (2000)
11. NVIDIA: CUDA C Programming Guide Version 5.5. available at `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`
12. Jordi, A., Wang, D.P.: sbpom: A parallel implementation of princenton ocean model. Environmental Modelling & Software 38, 59–61 (2012)
13. Yang, C., Xue, W., Fu, H., Gan, L., Li, L., Xu, Y., Lu, Y., Sun, J., Yang, G., Zheng, W.: A peta-scalable cpu-gpu algorithm for global atmospheric simulations. In: Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 1–12. ACM (2013)
14. Potluri, S., Wang, H., Bureddy, D., Singh, A.K., Rosales, C., Panda, D.K.: Optimizing mpi communication on multi-gpu systems using cuda inter-process communication. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), pp. 1848–1857. IEEE (2012)
15. Whitehead, N., Fit-Florea, A.: Precision & performance: Floating point and ieee 754 compliance for nvidia gpus. rn (A+ B) 21, 1–1874919424 (2011)
16. McCalpin, J., Wonnacott, D.: Time skewing: A value-based approach to optimizing for memory locality. Technical report, Technical Report DCS-TR-379, Department of Computer Science, Rugers University (1999)

Web Service Recommendation via Exploiting Temporal QoS Information

Chao Zhou, Wancai Zhang, and Bo Li

State Key Laboratory of Software Development Environment
School of Computer Science and Engineering, Beihang University, Beijing, China
{zhouchao,zhangwc,libo}@act.buaa.edu.cn

Abstract. With the rapid development of technologies based on Web service, a large quantity of Web services are available on the Internet. Web service recommendation aims at helping users in designing and developing service-oriented software systems. How to recommend web services with better QoS value receives a lot of attention. Previous works are usually based on the assumption that the QoS information is available. However, we usually encounter data sparsity issue, which demands the prediction of QoS Value. Also, the QoS Value of Web services may change over time due to the dynamic environment. How to handle the dynamic data streams of incoming service QoS value is a big challenge. To address above problems, we propose an Web Service Recommendation Framework by considering the temporal information. We explore to envision such QoS value data as a tensor and transform it into tensor factorization problem. A Tucker decomposition (TD) method is proposed to cope with the model which includes multidimensional information: user, service and time. To deal with the dynamic data streams of service QoS value, We introduce an incremental tensor factorization (ITF) method which is *scalable*, and *space efficient*. Comprehensive experiments are conducted on real-world Web service dataset and experimental results show that our approach exceed other approaches in efficiency and accuracy.

Keywords: Web Service, QoS, Recommendation.

1 Introduction

With the development of Service-oriented architecture (SOA), Web services have become standard software components deployed in the Internet. When developing service-oriented applications, designers try to find and reuse existing services to build the system business process. Meanwhile, many Web services are developed by different service providers with same function. Users pick over the services based on *Quality of Service* (QoS for short), such as cost, response time and availability [1].

Since selecting a Web service with high quality among a large number of candidates is a non-trivial task, effective approaches to service selection and recommendation are in urgent need. As the number of Web services with same function grows rapidly, the non-functional QoS properties, such as temporal or location context, become critical factors in Web service recommendation. Currently, many developers search services through public sites like Google Developers, prgrammableWeb, Yahoo! Pipes, etc. However, none of them provide temporal QoS information for users, and such information is quite important for software development. Due to the deployment mode and characteristics of Web services, the QoS inforamtion of Web services is greatly influenced by the service user's location and the invocation time.

In reality, the QoS data of Web services usually is sparse. Also, the QoS Value of Web services may change over time due to the dynamic environment and how to handle the dynamic data streams of incoming service QoS value is a big challenge. All these problems should be considered in Web service recommendation.

On the analysis of the existing problems, we propose a Web Service Recommendation Framework by way of exploiting temporal QoS information. The framework is comprised of an improved Collaborative Filtering algorithm incorporating the temporal information for QoS prediction and an incremental tensor factorization (ITF) method to deal with the dynamic data streams of service QoS value.

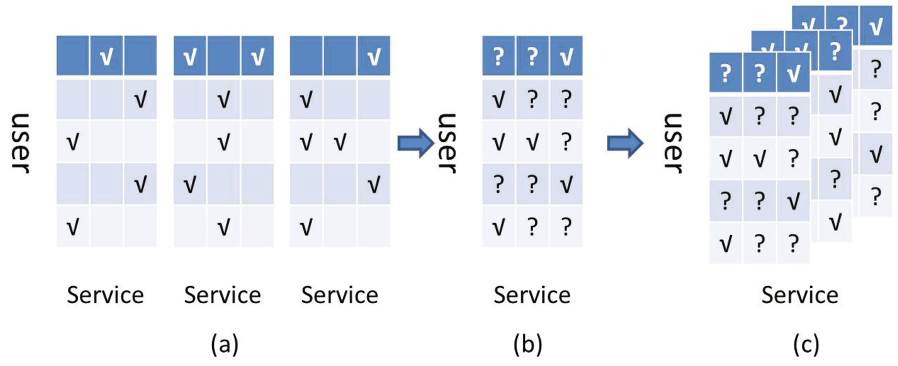


Fig. 1. Web service QoS information model

Firstly, when the Web service was once invoked, the Web service QoS information can be described by a two-dimensional *user-service* matrix. Service users may come from different locations and have little knowledge about each other, so it is quite normal they wouldnt share the QoS information with others. Consequently, as shown in Figure 1 (a), only a sub-fraction of the QoS information is available. To acquire sufficient information for Web service recommendation, the collaborative filtering methods[2,3,4,5,6] were adopted to predict the lacking QoS information. As shown in Figure 1 (b), the collaborative filtering method is basically based on the low-rank Matrix Factorization model [7]. In our work, the temporal information when the Web service was invoked is employed to improve the accuracy in predicting the QoS information. So we extend the two-dimensional QoS information to the triadic relation *user-service-time*. Intuitively, a tensor factorization [8] could be used to study the triadic relation in our problem. The triadic relation *user-service-time* was formalized as a tucker model [9] and the lacking QoS value was predicted by the tucker decomposition algorithm as depicted in Figure 1 (c).

Secondly, we introduce an incremental tensor factorization (ITF) method to deal with the dynamic data streams of service QoS value, which is represented as tensor stream using a sequence of multi-arrays. Algorithmically, we designed ITF very carefully so that it is *scalable* and *space efficient*.

In summary, this paper makes the following contributions:

1. The introduction of tensors to Web service missing QoS value prediction: the two-dimensional *user-service* matrix is extended into a *user-service-time* triadic relations represented by a three-dimensional tensor.

2. A Temporal QoS-Aware Web Service Recommendation Framework: in the framework, an Tucker decomposition method and an incremental tensor factorization algorithm are proposed to predict the QoS information on basis of the three-dimensional tensor of Web service Qos.

3. Systematic evaluation on large, real-world Web service QoS dataset, which is comprised of more than 150 millions records of Web service invocation from 408 service users on 5,473 Web services distributed on the Internet. This Web service dataset is available from[1].

2 Related Work

In previous years, Web Service Recommendation has attracted a lot of attention and lots of work have been done to improve Web service recommendation accuracy. Zeng et al. transformed the service selection to an optimization problem in [1]. The linear programming techniques are utilized in searching the optimal services. Alrifai et al. proposed a method incorporating global optimization with local selection to get an Web service candidate which is approximate to the optimal one in [10].

Recently, the collaborative filtering methods have been adopted by some works in Web Service Recommendation. Shao et al. [2] first used a user-based collaborative filtering algorithm in predicting the QoS of Web service. Due to the influence of user's geographic location on the prediction accuracy, Chen et al. [6] proposed a region based hybrid Collaborative Filter algorithm. Users are divided into regions according to the users' geographic locations. The algorithm will only searches the target user's region when similar users are identified, instead of searching the entire set. Zheng et al. [3] conducted extensive experiments based on real world dataset and designed a collaborative filtering algorithm by considering both the user and item information. Besides the geographic location information of users, Web services' location information is also included in the location-aware QoS prediction method In [11]. Zhang et al. [12] proposed that

[1] `http://www.service4all.org.cn/`

combining users' historical experiences, environmental factor and user input factor together to predict QoS value would achieve better accuracy. However, it was not mentioned about how to get access to environmental and user input factor. Lo et al. [5] extended the Matrix Factorization framework with relational regularization in the prediction of Web service QoS value for recommendation. The temporal information was considered in [13] for service recommendation, but it can't handle the dynamic data streams of service QoS value.

Those above works were mainly based on the two-dimensional Web service information of *user-service* . Due to the dynamic environment, the neglection of QoS information when the Web service was invoked may decrease the recommendation accuracy. Obviously different from those works, this paper's work also considers the temporal information when the Web service was invoked. So we extend the Web service static information model to dynamic triadic relations of *user-service-time* in Web service recommendation.

3 Model

3.1 Preliminaries

In this section, some basic definitions are given as the basis of the further introduction of tensor factorization.

Definition 1 ***(Tensor)*** *A tensor is an array with multi dimensions. The dimensionality of a tensor is defined as the number of dimensions.*

In our work, tensors are represented by letters $\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}}\cdots$; matrices by letters $\mathbf{A}$,$\mathbf{B}\cdots$; vectors by letters $\mathbf{a}$,$\mathbf{b}\cdots$; scalars by letters $a, b\cdots$. A tensor with N dimensions is represented as: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. In this tensor, the number of indices $(i_1, i_2, \cdots, i_N)$ is N and $a_{i_1 i_2\cdots i_N}$ represent the tensor's elements.

Definition 2 ***(a Tensor's Frobenius Norm)*** *The Frobenius norm of a tensor* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ *is defined as*

$$\|\boldsymbol{\mathcal{X}}\| = \sqrt{\langle\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{X}}\rangle} = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_N=1}^{I_N} x^2_{i_1 i_2\cdots i_N}}. \tag{1}$$

Based on the above definition, the inner product of two tensors $\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ can be specified as:

$$\langle\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}}\rangle = \sum_{i_1,i_2\cdots,i_N} a_{i_1 i_2\cdots i_N} b_{i_1 i_2\cdots i_N}.$$

Definition 3 ***(a Tensor's Rank)*** *The rank of a tensor* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, *rank*$(\boldsymbol{\mathcal{X}})$ *is the least integer such that there exist r rank-1 tensors whose sum is*

$$\boldsymbol{\mathcal{X}} = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \cdots \circ \boldsymbol{x}^{(N)}. \tag{2}$$

where $\circ$ *represents the vector outer product.*

Definition 4 ***(Tensor Matricization)*** *Tensor Matricization is transforming an array with N dimensions into a matrix.*

3.2 Problem Formulation

The QoS information of Web services will be gathered by the recommendation framework after a Web service was invoked by the end user. During a certain time, a large collection of QoS information will accumulate in the recommendation framework. The QoS information can be described as a set of quadruplets $\langle$*ServiceID, UserID, TimeID, Value*$\rangle$ (or $\langle u, s, t, v\rangle$ for short). *ServiceID, UserID, TimeID* represents the index according to the QoS *Value* of every Web service. Based on the Web service QoS information, a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ can be constituted as Figure 2, where I, J and K represent the number of *ServiceID*, *UserID* and *TimeID* respectively.

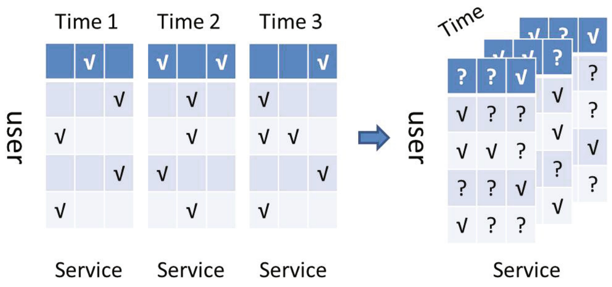


Fig. 2. Temporal Tensor Construct

A collection of quadruplets $\langle u, s, t, v\rangle$ from the Web service QoS dataset are used to construct the three-dimensional Tensor of Web service QoS model. The tensor can be represented as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ including the temporal information.

3.3 Offline Tensor Factorization

Definition 5 ***(Tucker Decomposition)*** *Given* $\boldsymbol{\mathcal{X}}$ *as a tensor of size* $O_1 \times O_2 \times \cdots \times O_N$ *and* $\hat{\boldsymbol{\mathcal{X}}}$ *be the approximation of tensor* $\boldsymbol{\mathcal{X}}$*. Tucker decomposition of* $\boldsymbol{\mathcal{X}}$ *outputs a tensor* $\boldsymbol{\mathcal{G}}$ *of size* $P_1 \times P_2 \times \cdots \times P_N$ *and factor matrixes* $\boldsymbol{A}^{(n)}$ *of size* $O_n \times P_n$ *. So, we can get*

$$\begin{aligned}\boldsymbol{\mathcal{X}} \approx \hat{\boldsymbol{\mathcal{X}}} &= \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \cdots \times_N \boldsymbol{A}^{(N)} = \boldsymbol{\mathcal{G}} \times \{\boldsymbol{A}\} \\ &= \sum_{i_1=1}^{P_1}\sum_{i_2=1}^{P_2}\cdots\sum_{i_N=1}^{P_N} \boldsymbol{\mathcal{G}}_{i_1 i_2\cdots i_N}\, \boldsymbol{a}_{i_1}^{(1)} \circ \boldsymbol{a}_{i_2}^{(2)} \circ \cdots \circ \boldsymbol{a}_{i_N}^{(N)}. \end{aligned} \tag{3}$$

A frequently-used approach for tucker decomposition is the alternating least squares (ALS for short) method. It computes the factor matrix one at a time while fixing other matrixes at the same time. In another word, the factor matrixes are fixed except $\mathbf{A}^{(n)}$ and computed as

$$\max_{\mathbf{A}^{(n)}} \left\| \boldsymbol{\mathcal{X}} \times \{\mathbf{A}^T\} \right\| . \tag{4}$$

Under the setting that

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{X}} \times_{-n} \{\mathbf{A}^T\}, \tag{5}$$

So the Equation (4) can be redefined as

$$\max_{\mathbf{A}^{(n)}} \left\| \mathbf{A}^{(n)^T} \mathbf{B}_{(n)} \right\| , \tag{6}$$

which is computed by the SVD of $\mathbf{B}_{(n)}$. More detail information is shown in[14].

In our work, the Web service QoS information is fomalized as a three-dimensional tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$. The tucker decomposition model is utilized to approximate the above QoS tensor. By studying the decomposed components, which is comprised of a tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R \times S \times T}$ and a series of factor matrixes: $\mathbf{U} \in \mathbb{R}^{I \times R}, \mathbf{V} \in \mathbb{R}^{J \times S}$ and $\mathbf{W} \in \mathbb{R}^{K \times T}$, the Web service QoS value of the specific time can be predicted. The QoS value $\hat{\boldsymbol{\mathcal{X}}}_{ijk}$ can be computed as:

$$\hat{\boldsymbol{\mathcal{X}}}_{ijk} = \sum_{r=1}^{R} \sum_{s=1}^{S} \sum_{t=1}^{T} \boldsymbol{\mathcal{G}}_{rst} \mathbf{u}_r \circ \mathbf{v}_s \circ \mathbf{w}_t . \tag{7}$$

Note that TD requires all tensors available up front, which is not possible for dynamic environments (i.e., new tensor keep coming). Even if we want to apply TD every time that a new tensor arrives, it is prohibitively expensive or merely impossible since the computation and space requirement are unbounded.

3.4 Incremental Tensor Factorization

Based on the incremental SVD [15], we presented below efficiently tensor factorization algorithm which is capable of incrementally updating when new data arrive. Given a 3-order tensor $\boldsymbol{\mathcal{A}}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^A}$, when a new 3-order tensor $\boldsymbol{\mathcal{F}}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^F}$ arrives, $\boldsymbol{\mathcal{A}}'$ is extended along the third order to form a tensor $\boldsymbol{\mathcal{A}} = (\boldsymbol{\mathcal{A}}' | \boldsymbol{\mathcal{F}}') \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ where the operation — merges the left an the right tensors along the third order, and $I_3 = I_3^A + I_3^F$. Figure 3 illustrates the process of unfolding $\boldsymbol{\mathcal{A}}$ and the relations between the previous unfolding matrices $\mathbf{A}'_{(1)}, \mathbf{A}'_{(2)}, \mathbf{A}'_{(3)}$, the new added unfolding matrices $\mathbf{F}'_{(1)}, \mathbf{F}'_{(2)}, \mathbf{F}'_{(3)}$ and the current unfolding matrices $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}, \mathbf{A}_{(3)}$. The three different modes of unfolding a extended 3-order tensor are shown in the left of Figure 3. The three unfolding matrices $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}$, and $\mathbf{A}_{(3)}$ corresponding to the three different modes are shown in the right of Figure 3.

After adding of the new tensor, the column vector of the two mode-1 and mode-2 unfolding matrices are extended, and the row vector of the mode-3 matrix is extended. Our incremental tensor factorization algorithm needs to online

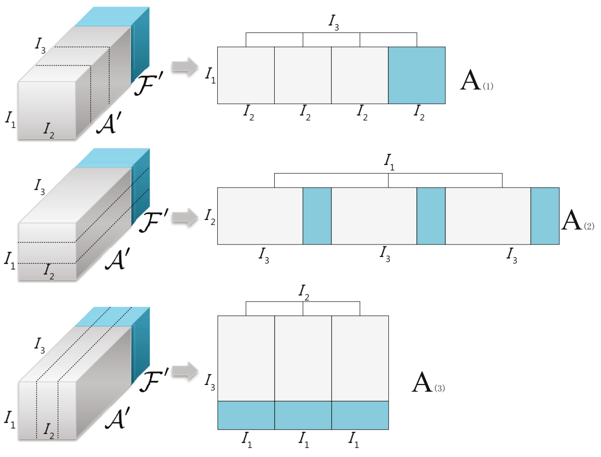


Fig. 3. Unfolding an extended 3-order tensor

track the changes in the three extended modes. The three modes are handled in the following ways:

1. As regards to $\mathbf{A}_{(1)}$, as $\mathbf{A}_{(1)} = (\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)})$, the SVD of $\mathbf{A}_{(1)}$ can be obtained from the SVD of $\mathbf{A}_{(1)}$ and $\mathbf{F}_{(1)}$ using the incremental SVD algorithm.

2. As regards to $\mathbf{A}_{(2)}$, it is noted that $\mathbf{A}_{(2)}$ can be decomposed as: $\mathbf{A}_{(2)} = (\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)}) \cdot \mathbf{P}$, where $\mathbf{P}$ is an orthonormal matrix obtained by column exchange and transpose operations on an identity matrix $\mathbf{Z}$ with rank $I_1 I_3$. Let

$$\mathbf{Z} = (\overbrace{\mathbf{E}_1}^{I_3^A} | \overbrace{\mathbf{Q}_1}^{I_3^F} | \overbrace{\mathbf{E}_2}^{I_3^A} | \overbrace{\mathbf{Q}_2}^{I_3^F} | \cdots | \overbrace{\mathbf{E}_{I_1}}^{I_3^A} | \overbrace{\mathbf{Q}_{I_1}}^{I_3^F})$$

which is generated by partitioning $\mathbf{Z}$ INTO $2I_1$ blocks along the column dimension. The partition of $\mathbf{Z}$ corresponds to $\mathbf{A}_2$ block partition shown in Figure 3 (i.e. $\mathbf{E}_1, \mathbf{E}_2, \cdots, \mathbf{E}_{I_1}$ correspond to the white regions, and $\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_{I_1}$ correspond to the gray regions. Consequently, the orthonormal matrix $\mathbf{P}$ is formulated as:

$$\mathbf{P} = (\mathbf{E}_1|\mathbf{E}_2|\cdots|\mathbf{E}_{I_1}|\mathbf{Q}_1|\mathbf{Q}_2|\cdots|\mathbf{Q}_{I_1})^T. \tag{8}$$

In this way, the $\mathbf{A}_{(2)}$ can be efficiently obtained by the SVD of $(\mathbf{A}'_{(2)}|\mathbf{F}'_{(2)})$ from the SVD of $\mathbf{A}'_{(1)}$ and $\mathbf{F}'_{(1)}$ using the incremental SVD algorithm.

3. As regards to $\mathbf{A}_{(3)}$, we estimate the row subspace, instead of the column subspace. We should calculate the SVD of the matrix $(\frac{\mathbf{A}'_{(3)}}{\mathbf{F}'_{(3)}})^T$, where - merges the upper and lower matrices. Due to $(\frac{\mathbf{A}'_{(3)}}{\mathbf{F}'_{(3)}})^T = ({\mathbf{A}'_{(3)}}^T|{\mathbf{F}'_{(3)}}^T)$, we can obtain the SVD of $\mathbf{A}_{(3)}$ from the SVD of ${\mathbf{A}'_{(3)}}^T$ and ${\mathbf{F}'_{(3)}}^T$ using the incremental SVD algorithm.

We can use the above results of $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}$ and $\mathbf{A}_{(3)}$ to formulate online 3-order factorization. In summary, Algorithm 1 presents the factorization scheme for online tensor factorization.

Algorithm 1. Incremental Tensor Factorization

Input: SVD of the mode-k unfolding matrix $\mathbf{A}_{(k)}$, i.e. $\mathbf{U}^{(k)}, \boldsymbol{\Sigma}_A^{(k)}, {\mathbf{V}^{(k)}}^T (1 \leq k \leq 3)$ of original tensor $\boldsymbol{\mathcal{A}}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^A}$ and new added tensor $\boldsymbol{\mathcal{F}}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^F}$.

Output: SVD of the mode-i unfolding matrix $\mathbf{A}_{(i)}$, i.e. $\hat{\mathbf{U}}^{(i)}, \hat{\boldsymbol{\Sigma}}_A^{(i)}, \hat{\mathbf{V}}^{(i)T} (1 \leq i \leq 3)$ of $\boldsymbol{\mathcal{A}} = (\boldsymbol{\mathcal{A}}' | \boldsymbol{\mathcal{F}}') \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ where $I_3 = I_3^A + I_3^F$
the approximate tensor $\hat{\boldsymbol{\mathcal{A}}}$.

```
1: $\mathbf{A}_{(1)} = (\mathbf{A}'_{(1)} | \mathbf{F}'_{(1)})$;
2: $\mathbf{A}_{(2)} = (\mathbf{A}'_{(1)} | \mathbf{F}'_{(1)}) \cdot \mathbf{P}$ where $\mathbf{P}$ is defined in Eq. (8);
3: $\mathbf{A}_{(3)} = (\frac{\mathbf{A}'_{(3)}}{\mathbf{F}'_{(3)}})^T$;
4: $\hat{\mathbf{U}}^{(1)} = \mathbf{U}^{(1)}, \hat{\boldsymbol{\Sigma}}_A^{(1)} = \boldsymbol{\Sigma}_A^{(1)}$ and $\hat{\mathbf{V}}^{(1)} = \mathbf{V}^{(1)}$;
5: $\hat{\mathbf{U}}^{(2)} = \mathbf{U}^{(2)}, \hat{\boldsymbol{\Sigma}}_A^{(2)} = \boldsymbol{\Sigma}_A^{(2)}$ and $\hat{\mathbf{V}}^{(2)} = \mathbf{P}^T \mathbf{V}^{(2)}$;
6: $\hat{\mathbf{U}}^{(3)} = \mathbf{V}^{(3)}, \hat{\boldsymbol{\Sigma}}_A^{(3)} = {\boldsymbol{\Sigma}_A^{(3)}}^T$ and $\hat{\mathbf{V}}^{(3)} = \mathbf{U}^{(3)}$.
```

4 Experiments

In the following part, the dataset, metric and the experiment results including accuracy and efficiency comparison are introduced. The TD and ITF algorithms were implemented in Matlab 2008. We conducted the experiments on a Dell server of PowerEdge T620. The server has processors with Intel Xeon 2.00GHz and RAM with 16GB DDR3 1600. The operation system is Window Server 2008.

4.1 Dataset

In order to validate the accuracy and efficiency of the approach proposed in this paper, two prototype systems were developed, including an QoS detection middleware *QoSDetecter* and a Web service invocation system *ServiceXChange*. In *ServiceXChange*, we have collected more than 20,000 records of accessible Web services by searching on the Internet and integrated the platform with *Service4All*[2] which is an comprehensive development environment for Service-oriented software. We developed a united interface to invoke these Web services and designed an *xml* file to adjust to invoke the corresponding method automatically.

Obtaining the QoS information of Web services used by users from distributed locations is a tough work. In order to observe and gather QoS information of

[2] `http://www.service4all.org.cn/`

the Web services, we adopted the distributed nodes from Planet-Lab[3]. Planet-Lab includes more than 1,000 nodes at over 500 sites, constituting a world-wide research network on the Internet. More than 600 nodes from distributed locations on the Planet-Lab were employed to keep track of the QoS information of Web services while a series of invocation on specified Web services were conducted on the nodes.

When the raw data of QoS information were processed, 408 nodes on the Planet-Lab were appointed as the simulative service users and 5,473 openly accesible real-world Web services were invoked and kept track of by every node for a period of time. At the end, we collected Web services QoS information between 30 days from October 14 and November 21 of 2013 in the dataset.

In this dataset, about 37 million records of quadruplets $\langle u, s, t, v\rangle$ are available and two $408 \times 5473 \times 240$ *user-service-time* tensors can be constructed from this dataset. Every tensor is consisted of Web service QoS information: response time and throughput. Web service response time represents the time interval between when a request was sent to the Web service and when the relevant response was received by the end service user. Web service throughput represents the amount of requests it can deal with in a particular period of time. Table 1 shows the detailed statistics of Web service QoS information in the dataset. Briefly, we adopted response time and throughput as the QoS evaluation criteria in our work. In general, the approach proposed in this paper can be applied in other QoS attributes.

Table 1. Web Service QoS Information Dataset Statistics

Statistics	Response Time	Throughput
Scale	0-200s	0-1000kbps
Mean value	0.6033	8.2107
Service users number	408	408
Web services number	5473	5473
Time periods	240	240

4.2 Metrics

In order to measure the accuracy of the approaches proposed in our work, we adopted *Mean Absolute Error* (MAE) [4] as the metric and compared the results with other collaborative filter methods. MAE is formalized as:

$$MAE = \frac{1}{N} \sum_{i,j,k} \left| \boldsymbol{\mathcal{X}}_{ijk} - \hat{\boldsymbol{\mathcal{X}}}_{ijk} \right|$$

where $\boldsymbol{\mathcal{X}}_{ijk}$ represents the real QoS information of Web services j obtained by user i at given time k; $\hat{\boldsymbol{\mathcal{X}}}_{ijk}$ represents the output of the QoS prediction algorithm; N represents the number of Web services.

[3] `http://www.planet-lab.org/`

4.3 Baselines

Based on the dataset, we study the method TD and ITF proposed in our work under the following baselines:

- *UMean*: The average value of QoS information from each user is used in this method.
- *IMean*: The average value of QoS information of every Web service is used in this method.
- *WSRec*: A hybrid collaborative filtering algorithm based on similar users and Web services is used in this method[3].
- *RSVD*: Singular Value Decomposition (SVD for short) is employed to mine the '*latent structure*' of the raw data. In this paper, the regularized SVD in [16] is used as the baseline algorithm.

4.4 Accuracy Comparison

Accuracy is an important indicator in Web service recommendation. Those 4 baseline algorithms cannot be applied in the context of QoS prediction based on three-dimensional QoS information directly, so we transform the triadic relation of *user-service-time* to a series of *user-service* matrixes in accordance with time slices. After that, the *user-service-time* tensor becomes a compact *user-service* matrix. Every element of the matrix can be calculated by the mean value of the specified pair of $\langle user, service \rangle$ in all the time slices. For every matrix, the above 4 baseline algorithms are employed to predict the Web service QoS value. Finally, the mean absolute error of every baseline algorithm is computed and compared with the TD and ITF algorithm.

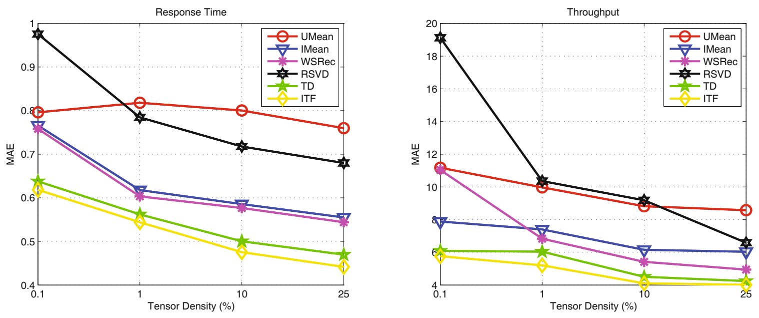


Fig. 4. Impact of Sparseness

The data sparsity of QoS information has great influence on the accuracy of QoS prediction. The density of the tensors is set as 0.1%, 1%, 10% or 25%. Figure 4 shows the results of Web service response time and throughput under different algorithms. From figure 4 , we can see that: the performance of the TD and ITF algorithms strengthens as the tensor density increases; the TD and ITF algorithms outperform other baseline algorithms under all circumstances. According to our analysis, sufficient QoS information will result in better prediction accuracy and the neglection of temporal information is the main reason why these baseline algorithms failed.

4.5 Efficiency Comparison

In order to evaluate the efficiency of the TD and ITF algorithms in QoS prediction, we set a series of data streams as tensors with length of 10, 50, 100 and 240. Because both algorithms *UMean* and *IMean* simply compute the average value of QoS, TD, ITF, *WSRec* and *RSVD* are selected for comparison in this section. Experiment results from Figure 5 shows how tensors with different length influence the execution efficiency while the dimensionality is set as 30. As the stream data length increases, the ITF performs better than other algorithms all the time.

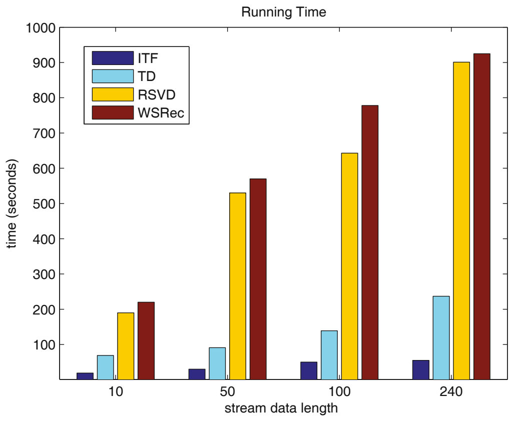


Fig. 5. Efficiency Comparison

5 Conclusions

In conclusion, we extend the two dimension static QoS informaion to three dimensions by taking into account the time context when the Web service were invoked and propose the tensor factorization method to improve the accuracy and efficiency of Web service recommendation. A Tucker decomposition method is proposed to cope with the tensor model and an incremental tensor factorization algorithm is designed to deal with the dynamic data streams of service QoS value. A comprehensive Web service recommendation framework is designed and extensive experiments are conducted on the real-world Web services. As shown in the experimental results, when compared with other Web service recommendation methods, our work achieves better performance in the aspects of efficiency and accuracy.

Web services are deployed on the Internet which is changing dynamically all the time. In the future work, additional Web services information under different context will be gathered as the base of Web service recommendation. Also, the various methods should be synthesized to improve the recommendation effect.

Acknowledgement. The authors gratefully acknowledge the anonymous reviewers for their helpful suggestions and insightful comments to improve the quality of the paper. The work reported in this paper has been partially supported by National Nature Science Foundation of China (No. 61202424), State Key Laboratory of Software Development Environment (No. SKLSDE-2013ZX-32).

References

1. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: Proceedings of the 12th International Conference on World Wide Web, pp. 411–421. ACM (2003)
2. Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei, H.: Personalized qos prediction forweb services via collaborative filtering. In: IEEE International Conference on Web Services ICWS 2007, pp. 439–446 (2007)
3. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Wsrec: A collaborative filtering based web service recommender system. In: 2009 IEEE International Conference on Web Services, pp. 437–444 (2009)
4. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service qos prediction via neighborhood integrated matrix factorization. IEEE Transactions on Services Computing 6(3), 289–299 (2013)
5. Lo, W., Yin, J., Deng, S., Li, Y., Wu, Z.: Collaborative web service qos prediction with location-based regularization. In: Proceedings of the 2012 IEEE 19th International Conference on Web Services, pp. 464–471. IEEE (2012)
6. Chen, X., Liu, X., Huang, Z., Sun, H.: Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: 2010 IEEE International Conference on Web Services (ICWS), pp. 9–16. IEEE (2010)
7. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
8. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review 51(3), 455–500 (2009)
9. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31(3), 279–311 (1966)
10. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web, pp. 881–890. ACM (2009)
11. Tang, M., Jiang, Y., Liu, J., Liu, X.: Location-aware collaborative filtering for qos-based service recommendation. In: 2012 IEEE 19th International Conference on Web Services (ICWS), pp. 202–209. IEEE (2012)

12. Li, Z., Bin, Z., Ying, L., Yan, G., Zhi-Liang, Z.: A web service qos prediction approach based on collaborative filtering. In: 2010 IEEE Asia-Pacific Services Computing Conference (APSCC). IEEE, pp. 725–731 (2010)
13. Zhang, W., Sun, H., Liu, X., Guo, X.: Temporal qos-aware web service recommendation via non-negative tensor factorization. In: Proceedings of the 23rd International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 585–596 (2014)
14. Kolda, T.G.: Multilinear operators for higher-order decompositions. United States. Department of Energy (2006)
15. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Fifth International Conference on Computer and Information Science, pp. 27–28. Citeseer (2002)
16. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD Cup and Workshop, vol. 2007, pp. 5–8 (2007)

Optimizing and Scaling HPCG on Tianhe-2: Early Experience

Xianyi Zhang[1,3], Chao Yang[1,2], Fangfang Liu[1], Yiqun Liu[1,3], and Yutong Lu[4]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[2] State Key Laboratory of Computer Science, Chinese Academy of Sciences, Beijing 100190, China
[3] University of Chinese Academy of Sciences, Beijing 100049, China
[4] National University of Defense Technology, Changsha, Hunan 410073, China

Abstract. In this paper, a first attempt has been made on optimizing and scaling HPCG on the world's largest supercomputer, Tianhe-2. This early work focuses on the optimization of the CPU code without using the Intel Xeon Phi coprocessors. In our work, we reformulate the basic CG algorithm to minimize the cost of collective communication and employ several optimizing techniques such as SIMDization, loop unrolling, forward and backward sweep fusion, OpenMP parallization to further enhance the performance of kernels such as the sparse matrix vector multiplication, the symmetric Gauss–Seidel relaxation and the geometric multigrid v-cycle. We successfully scale the HPCG code from 256 up to 6,144 nodes (147,456 CPU cores) on Tianhe-2, with a nearly ideal weak scalability and an aggregate performance of 79.83 Tflops, which is 6.38X higher than the reference implementation.

1 Introduction

The High Performance Conjugate Gradient (HPCG) benchmark [2,3], recently introduced to the HPC world, aims to better correlate computation and data access patterns found in many applications nowadays. Compared to the High Performance Linpack (HPL), which is the de-facto standard to rank supercomputing systems on the TOP500 List, HPCG contains both sparse matrix kernels that are highly bandwidth limited and collective communications that are difficult to scale. Optimizing HPCG is a challenging task for a high computing throughput, yet low data-moving bandwidth system popularized on today's TOP500 List.

In this work, we shall make a first attempt to optimize and scale HPCG on the Tianhe-2 supercomputer. Announced in June 2013 and undertaking the top spot on the TOP500 List since then, Tianhe-2 is the largest heterogeneous system in the world. It is comprised of both Intel Xeon processors and Intel Xeon Phi coprocessors. This early work only focuses on optimizing HPCG on the CPU processors without using the coprocessors. The primary contributions of our work include:

– A reformulated but mathematically equivalent variant of the Conjugate Gradient algorithm is employed to reduce the cost of global communication.

By using the reformulated algorithm, the three vector dot-products in each iteration can be done simultaneously with only one nonblocking collective communication that can be further overlapped with other operations.
- A simplified SELLPACK sparse matrix storage format is chosen to replace the default CSR format in both the sparse matrix vector multiplication and the symmetric Gauss–Seidel kernels. By using the new format, the locality of accessing dense vectors is greatly improved the per-core performance is substantially increased.
- A variety of optimization techniques such as vectorization, loop unrolling, forward and backward fusion, division optimization, and OpenMP parallization are carefully exploited. The optimizations improve the single-CPU performance of HPCG to about 6.5X of the reference on an Intel Sandy Bridge platform.
- The optimized parallel code scales up to 6144 nodes (147.4 thousands CPU cores) with a nearly ideal weak-scaling efficiency of 96.96% and an aggregated performance of 79.83Tflops on Tianhe-2.

We outline the paper as follows. In Section 2 the basic HPCG algorithm is provided, together with an explanation on optimizations permitted by HPCG. We then present in Section 3 the reformulated algorithm together with the details of implementations and optimizations. In Section 4 we show the performance results on both an Intel Sandy Bridge platform and Tianhe-2, following which comments on related works are given in Section 5. The paper is concluded in Section 6.

2 Basic HPCG Algorithms

The High Performance Conjugate Gradient (HPCG), proposed in 2013 [2], is a benchmark program that solves a sparse linear system arising in solving a three-dimensional heat diffusion problem. Our research is based on the latest available version of HPCG, v2.1.

As explained in its technical specification [3], HPCG intends to solve the linear system generated from the finite difference discretization of the Poisson equation: $-\Delta u = f$, with homogeneous Dirichlet boundary conditions applied along the boundary of a three-dimensional cubic domain Ω. Based on a semistructured mesh with equidistant mesh spaces in the x, y and z directions, respectively, the discretization employed in HPCG leads to a second-order accurate 27-point stencil as shown in Fig. 1. For distributed parallel computing, HPCG makes use of a three-dimensional domain decomposition strategy and assigns one MPI process to each of the resulted small subdomains.

The sparse linear system solved in HPCG is $Ax = b$. Here A is a sparse matrix obtained from the 27-point discretization of the Poisson equation, b is the right-hand-side vector generated by using an exact solution with all one's. In HPCG, sparse matrices and vectors are stored distributively among MPI processes corresponding to the decomposed subdomains. It is allowed to modify the data formats of both sparse matrices and vectors in HPCG to achieve better

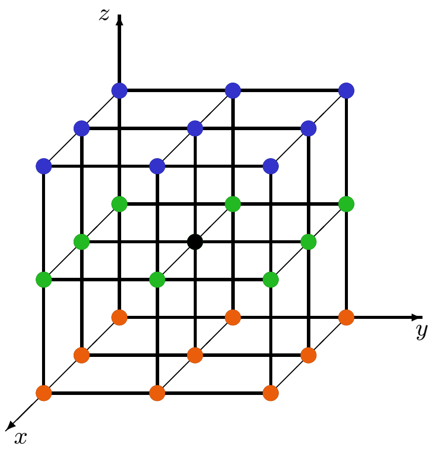


Fig. 1. 3D 27-point stencil in HPCG

performance, but it is prohibited to take advantage of the specific structure of the sparse matrix (such as the diagonal structure, the specific values of nonzero entries, etc), nor is it permitted to replace sparse matrices with stencil computations.

In HPCG, the linear system $Ax = b$ is solved by using a Preconditioned Conjugate Gradient (PCG) algorithm as described in Algorithm 1, where M^{-1} is the multigrid preconditioner to be elaborated later. In the main loop of Algorithm 1, the major cost consists of one sparse matrix vector multiplication (SpMV) in line 7, one preconditioner application (MG) in line 3, three vector dot-products in line 4, 7 and 10, and three vector updates (AXPBY) in line 6, 8 and 9.

Algorithm 1. PCG for $Ax = b$

```
Input: $A$, $b$, $x_0$, $it_{max}$, $\varepsilon$
 1: $r_0 \leftarrow b - Ax_0$
 2: repeat $(i = 0, 1, ...)$
 3:     $z_i \leftarrow M^{-1} r_i$
 4:     $s_i \leftarrow (r_i, z_i)$
 5:     if $(i = 0)$  $p_i \leftarrow z_i$
 6:     else          $p_i \leftarrow z_i + (s_i/s_{i-1}) p_{i-1}$
 7:     $\alpha_i \leftarrow s_i/(p_i, Ap_i)$
 8:     $x_{i+1} \leftarrow x_i + \alpha_i p_i$
 9:     $r_{i+1} \leftarrow r_i - \alpha_i A p_i$
10: until $(i + 1 = it_{max})$ or $(||r_{i+1}||_2/||r_0||_2 \leq \varepsilon)$
Output: $x_{i+1}$
```

The preconditioner M^{-1} employed in HPCG is based on a V-cycle geometric multigrid method, which is depicted in Fig. 2. In HPCG, the total number of levels of the V-cycle is fixed to 4 (including the finest level itself). At each level,

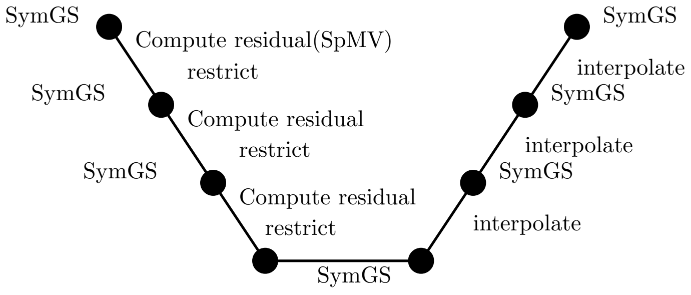


Fig. 2. The V-cycle geometric multigrid algorithm

a symmetric Gauss–Seidel algorithm is used as both smoothers (including pre- and post-smoothing) and coarse level solver. Only one step of the symmetric Gauss–Seidel iteration is applied at each level. Between levels, a simple injection operator and its transpose are used as the intergrid operators (i.e., restrictions and prolongations).

3 Implementation and Optimization

In the standard PCG algorithm, three vector dot-products need to be done per iteration, with each one requiring a global collective communication that may substantially degrade the scalability at scale. In order to reduce the global communication overhead, we employ a reformulated but mathematically equivalent variant of the basic PCG algorithm, the pipelined PCG [5].

Algorithm 2. Pipelined PCG for $Ax = b$

```
Input: $A$, $b$, $x_0$, $it_{max}$, $\varepsilon$
 1: $r_0 \leftarrow b - Ax_0,\ u_0 \leftarrow M^{-1}r_0,\ w_0 \leftarrow Au_0$
 2: repeat $(i = 0, 1, ...)$
 3:     $\gamma_i \leftarrow (r_i, u_i)$
 4:     $\delta_i \leftarrow (w_i, u_i)$
 5:     $m_i \leftarrow M^{-1}w_i$
 6:     $n_i \leftarrow Am_i$
 7:     if $(i = 0)$  $\beta_i \leftarrow 0,$        $\alpha_i \leftarrow \gamma_i/\delta$
 8:     else          $\beta_i \leftarrow \gamma_i/\gamma_{i-1},\ \alpha_i \leftarrow \gamma_i/(\delta - \beta_i\gamma_i/\alpha_{i-1})$
 9:     $s_i \leftarrow w_i + \beta_i s_{i-1},\ r_{i+1} \leftarrow r_i - \alpha_i s_i$
10:     $p_i \leftarrow u_i + \beta_i p_{i-1},\ x_{i+1} \leftarrow x_i + \alpha_i p_i$
11:     $q_i \leftarrow m_i + \beta_i q_{i-1},\ u_{i+1} \leftarrow u_i - \alpha_i q_i$
12:     $z_i \leftarrow n_i + \beta_i z_{i-1},\ w_{i+1} \leftarrow w_i - \alpha_i z_i$
13: until $(i + 1 = it_{max})$ or $(||r_{i+1}||_2/||r_0||_2 \leq \varepsilon)$
Output: $x_{i+1}$
```

As shown in Algorithm 2, the pipelined PCG method has two advantages. First, only one global reduction is required for each iteration. Second, the global reduction can be overlapped with the matrix-vector product and with the application of the preconditioner. However, there are also some price to pay. An extra application of SpMV and preconditioner is required before starting the iteration loop. In addition, by using the new algorithm, the number of AXPBY's in each iteration has increased from 3 to 8. The overhead introduced by the extra AXPBY's can be reduced by properly changing the calling sequence and fusing the input and output vectors, as shown in line 9-12 in Algorithm 2.

3.1 Sparse Matrix Storage Format

The sparse matrix storage format plays a key role in SpMV and SymGS performance. In the HPCG reference implementation, a standard Compressed Sparse Row (CSR) format is used. Although CSR is a wildly used format, it is unable to exploit the feature in matrices generated by HPCG.

$$A = \begin{pmatrix} 1\,0\,0\,5 \\ 0\,2\,6\,0 \\ 7\,0\,3\,0 \\ 0\,8\,9\,4 \end{pmatrix} \quad \begin{array}{l} nnz = \begin{bmatrix} 1\,2\,5\,6 * * 7\,8\,3\,9 * 4 \end{bmatrix} \\ cols = \begin{bmatrix} 0\,1\,3\,2 * * 0\,1\,2\,2 * 3 \end{bmatrix} \end{array} \tag{1}$$

We make use of a simplified SELLPACK (Sliced ELLPACK) storage format [7], which is a variant of ELLPACK. The SELLPACK format splits nb rows into a block and stores the column major in each block. Because the number of nonzeros per row is mostly 27 in HPCG matrices, we fix the number of nonzeros per row for every block in the SELLPACK format. Equation (1) shows an example for the simplified SELLPACK with $nb = 2$. The nnz array stores nonzero values and the $cols$ array stores the column index. In the example above, the number of nonzeros per row is 3 and the asterisk indicates the padding element.

The reason we chose to use the SELLPACK format is better locality for accessing the dense input vector x in SpMV and SymGS. According to some experiments, we find $nb = 8$ leads to optimal performance among 4, 8 and 16.

3.2 SpMV

A native implementation of SpMV for the simplified SELLPACK format is depicted in Algorithm 3. We optimize the performance of SpMV from the following aspects.

SIMDization. We use SIMD intrinsic functions to accelerate the innermost loop j, i.e., line 4-7 in Algorithm 3. For $nb = 8$ in our case, we may totally unroll the loop j. It is trivial to replace scalar instructions with SIMD instructions for sum and nnz. However, it is difficult to vectorize the load of vector x without SIMD gather instructions, which is only available on AVX2 and AVX-512 instruction sets. Therefore, we use scalar instructions to access the column index array $cols$ and the vector x. Then, we compose each individual value of vector x to SIMD vectors. On top of that, we can use SIMD multiplication and addition instructions for line 6 in Algorithm 3.

Algorithm 3. Simplified SELLPACK SpMV native implementation

```
Input: $A(nnz, cols, nrows, nb, nnzs\_in\_row)$, $x$, $y$
 1: for $i = 0, \ldots, nrows/nb$ do
 2:     $sum[0 : nb] = 0.0$
 3:     for $k = 0, \ldots, nnzs\_in\_row$ do
 4:         for $j = 0, \ldots, nb$ do
 5:             $curCol := cols[i \times (nb \times nnzs\_in\_row) + j + k \times nb]$
 6:             $sum[j] += nnz[i \times (nb \times nnzs\_in\_row) + j + k \times nb] \times x[curCol]$
 7:         end for
 8:     end for
 9:     $y[i \times nb : (i + 1) \times nb] = sum[0 : nb]$
10: end for
Output: $y = Ax$
```

Loop Unrolling. Since the innermost loop is totally unrolled, we may further unroll loop k at line 3 in Algorithm 3. After testing unrolled factor 2, 4, 8, and 16, we find that factor 4 is the optimal choice.

Parallelization. We distribute the *nrows* block among CPU threads via OpenMP compiler directives on loop i at line 1 in Algorithm 3. Because the compute is independent on every row, it does not need extra synchronization operations. Since the number of nonzeros is same in every block, the workload is almost balanced. Thus, we may choose the `static` schedule strategy in OpenMP.

3.3 SymGS

Shown in Algorithm 4 is a native implementation of SymGS based on the simplified SELLPACK format. In the algorithm, line 1-11 is the forward sweep and line 12-22 is the backward sweep. Besides the different accessing order, the forward and backward sweeps contain similar computing operations. We optimize the performance of SymGS in the following way.

SIMDization and Loop Unrolling. Because the original SymGS contains data dependence between the current and the previous iterations, it is impossible to vectorize the innermost loop. Thus, we use a point-wise relaxation in every *nb* block, which is similar to Garcia's work on vectorizing multigrid [4]. Analogous to the SpMV vectorization, we use SIMD instructions to load *nnz* and r. Meanwhile, we compose x values to a vector register. Then, we compute the result by using SIMD multiplication, addition, and division instructions and finally write back to vector x. After the SIMDization of the innermost loop, we may further unroll the loop in every *nb* block, which is similar to the case of SpMV optimization. We also chose 4 as the unrolled factor based on some experiments.

Algorithm 4. Simplified SELLPACK SymGS native implementation

```
Input: $A$ ($nnz$, $cols$, $nrows$, $nb$, $nnzs\_in\_row$), $x$, $r$
 1: for $i = 0, \ldots, nrows$ do
 2:     $sum{:=}r[i]$
 3:     $block\_id := i/nb$
 4:     $id\_in\_block := i\%nb$
 5:     for $k = 0, \ldots, nnz\_in\_row$ do
 6:         $curCol := cols[block\_id \times (nb \times nnzs\_in\_row) + id\_in\_block + k \times nb]$
 7:         $sum[j]$ -= $nnz[block\_id{\times}(nb{\times}nnzs\_in\_row){+}id\_in\_block{+}k{\times}nb]{\times}x[curCol]$
 8:     end for
 9:     $sum$ += $x[i] \times diagonal[i]$
10:     $x[i] := sum/diagonal[i]$
11: end for
12: for $i = nrows - 1, \ldots, 0$ do
13:     $sum := r[i]$
14:     $block\_id := i/nb$
15:     $id\_in\_block := i\%nb$
16:     for $k = 0, \ldots, nnz\_in\_row$ do
17:         $curCol := cols[block\_id \times (nb \times nnzs\_in\_row) + id\_in\_block + k \times nb]$
18:         $sum[j]$ -= $nnz[block\_id{\times}(nb{\times}nnzs\_in\_row){+}id\_in\_block{+}k{\times}nb]{\times}x[curCol]$
19:     end for
20:     $sum$ += $x[i] \times diagonal[i]$
21:     $x[i] := sum/diagonal[i]$
22: end for
Output: $SymGS(A, x, r)$
```

Division Optimization. Because division instructions are not pipelined, we need to replace them with other pipelined instructions. For the integer division (in line 3 and 14 in Algorithm 4), we can replace it by the bitwise right shift instruction since nb is the power of 2. For the floating-point division (line 10 and 21 in Algorithm 4), the scalar native implementation can get some benefit by setting the `-no-prec-div` compiler flag. However, after the vectorization optimization, we use the SIMD division instruction manually. Thus, we need to add an extra array to store the reciprocal of diagonals and convert the SIMD division instruction to SIMD multiplications.

Fused Forward-Backward Sweep. In the native implementation of SymGS, the sparse matrix A has been traversed twice, of which the first one is by the forward sweep in a certain order and the second by the backward sweep in the reverted order. It is therefore possible to improve the cache locality by fusing the forward and backward sweeps. Firstly, we use a red-black relaxation in z axis as shown in Fig. 3. For this case, there are four planets. In the forward sweep, z_0 and z_2 are red planets and z_1 and z_3 are black planets. In backward sweep, z_3 and z_1 are red planets and z_2 and z_0 are black planets. Next, we combine the forward and backward sweeps in the following four steps.

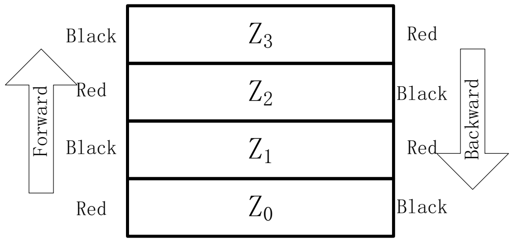


Fig. 3. An SymGS example for red-black coloring along axis z

1. Compute forward sweeps of z_0 and z_2 red planets.
2. Compute forward sweep of z_1 black planet.
3. Compute backward sweep of z_1 red planet.
4. Compute backward sweep of z_0 black planet.

We present in Algorithm 5 the detail of fused forward-backward sweep in SymGS. In the fused algorithm, line 2-7 is the main loop and the fused the sweep, while line 8-10 deals with the tail along z axis.

Algorithm 5. Fused forward-backward sweep in SymGS

```
1: Forward sweep along $z_0$
2: for $z_i = 2, \ldots, nz$, Step 2 do
3:     Forward sweep along $z_i$
4:     Forward sweep along $z_{i-1}$
5:     Backward sweep along $z_{i-1}$
6:     Backward sweep along $z_{i-2}$
7: end for
8: Forward sweep along $z_{nz-1}$
9: Backward sweep along $z_{nz-1}$
10: Backward sweep along $z_{nz-2}$
```

Parallelization. We parallelize Algorithm 5 by using OpenMP. Because there exists data dependence between the planets, we split along the y axis and employ another level of red-black relaxation in each planet as shown in Fig. 4. Every thread performs computation in a number of successive columns. Between red and black computations, we insert an explicit OpenMP barrier.

3.4 Pipelined PCG

In Algorithm 2, there are two dot-products in each iteration. In addition to that, the calculation of the residual (i.e., the norm of r) is required to check the convergence. Therefore, the algorithm needs three global reductions in every iteration. We fuse the three global reductions to one and overlap it with the matrix-vector product and the application of the preconditioner. This can be

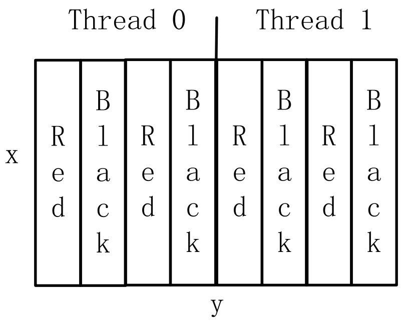


Fig. 4. An example of parallel SymGS in planet

easily implemented by non-blocking communication, which is available in the MPI-3 standard. In our implementation, we start a non-blocking reduction by `MPI_Iallreduce` and then proceed with the SpMV and MG. After both SpMV and MG are done, we use `MPI_Waitall` to finish the reduction. Furthermore, in order to decrease the memory access overhead, we rearrange the sequence of AXPBY's and merge each two operations that use a same vector. A new subroutine is defined to replace each two consecutive XAPBY's.

3.5 Other Optimizations

Analogous to other works on optimizing geometric multigrid kernels such as the one in [11], we fuse the residual computation (SpMV) and the restriction operation to a single subroutine in the V-cycle. This subroutine eliminates the residual computations that are unused on the coarse grid. In order to further enhance the performance, compiler directives are utilized to vectorize and parallelize the vector operations including vector AXPBY's, dot-products, prolongations.

4 Performance and Analysis

We optimized HPCG 2.1 version. Two platforms are employed for the performance tests. The first one is an Intel Sandy Bridge platform that is mainly used to test the single-CPU performance by exploiting various optimization techniques. The second testbed is the Tianhe-2 supercomputer, which is used for large-scale tests. The configurations of both platforms are shown in Table 1. We use the Intel Sandy Bridge computer for single-CPU performance analysis due to the limited access of Tianhe-2. Although the two computers are different, they belong to similar microarchitectures (Sandy Bridge and Ivy Bridge).

4.1 Single-CPU Performance

We conduct the single-CPU performance tests on the Intel Sandy Bridge computer based on three typical data sizes, including 32×32×32, 64×64×64, and

Table 1. Configurations of the Intel Sandy Bridge platform and Tianhe-2

Platform	Intel Sandy Bridge	Tianhe-2 (Intel Ivy Bridge)
CPU	Intel Xeon E5-2680@2.7GHz	Intel Xeon E5-2692 v2@2.2GHz
Cores/CPU	8	12
CPUs/Node	2	2
Memory/Node	32GB	32GB
OS	Linux 3.5.0	Linux 2.6.32
Compiler	icpc 14.0.0	icpc 14.0.0
Compiler Options	-O3 -openmp -xHOST -no-prec-div	-O3 -openmp -xAVX -no-prec-div
MPI	Intel MPI 4.1.3	Customized MPICH2-3.04

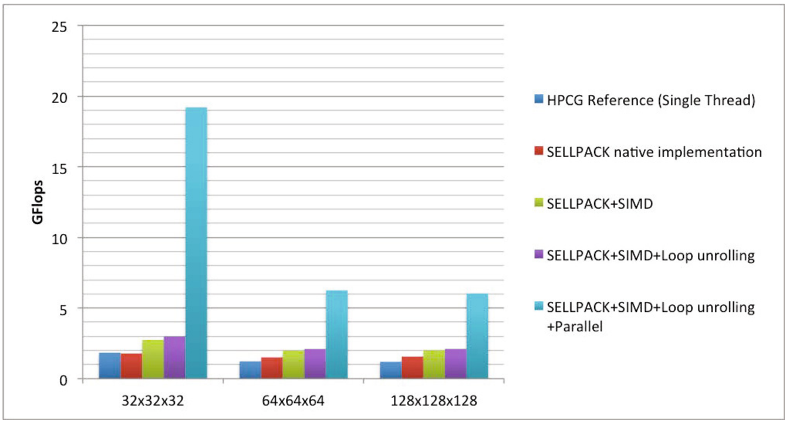


Fig. 5. The performance of SpMV on a single Intel Sandy Bridge CPU (8 cores)

128×128×128. The performance results of SpMV are shown in Fig. 5. From the results we observe:

- The performance of SpMV has a strong correlation to both the data size and the cache capacity. The small data size of 32×32×32 achieves the best performance of around 2.99GFlops on single core and 19.20GFlops on eight cores. The reason is that the working set is small enough to fit into the L3 cache. For larger data sizes such as 64×64×64 and 128×128×128, the performance drops to 6.1GFlops on eight cores because the data size exceeds the cache limit.
- By employing the SELLPACK storage format, we obtain 22% performance gain on both 64×64×64 and 128×128×128 cases. The performance boost indicates that better locality is exploited for the sparse matrices. For the small size 32×32×32, the SELLPACK format is unable to improve the performance since the data is already located in cache.
- On single core, the SIMDization offers the highest performance, which surpasses the scalar code by over 30%. The loop unrolling technique only improves the performace by about 5%.

– The parallel speedup was about 3.0X for eight cores when the data size is large. The effective bandwidth of our SpMV implementation achieves 37GB/s, which is very close to the STREAM bandwidth. This indicates that the optimization is nearly optimal.

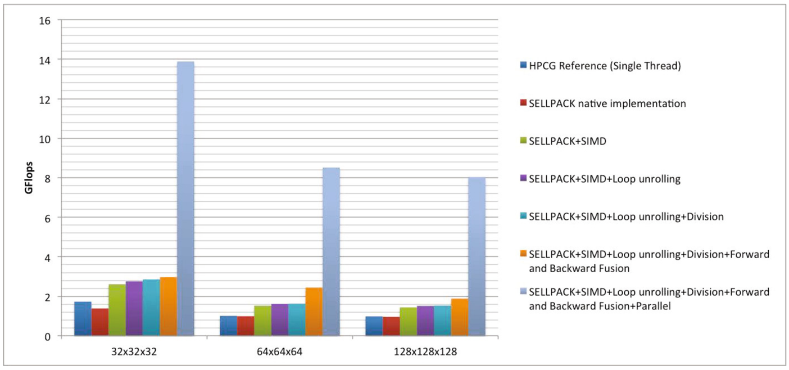


Fig. 6. The performance of finest-level SymGS on a single Intel Sandy Bridge CPU (8 cores)

The performance of SymGS on the finest-level is presented in Fig. 6, from which we find:

– Analogous to the SpMV results, the case of 32×32×32 data size also achieves the highest performance, which is 2.97GFlops on single core and is about 13.75% of the peak performance.
– Because of the lower memory accessing locality of the SELLPACK scalar implementation, the performance is even worse than the reference implementation. However, the vectorization improves the performance by over 50%. The performance gain of loop unrolling is about 5%-10%.
– The division optimization can improve the performance by about 10% for the 32×32×32 case. For other cases, we only obtains about 1% of performance increase. The reason is that the extra array consumes the limited cache.
– The forward-backward sweep fusion improves the performance by 5%, 50% and 20% respectively. Because the data size of 32×32×32 can fit into the L3 cache, the improvement is lower than other cases.
– The parallel speedups are about 4.64X, 3.48X, and 4.20X on eight cores. For large cases, the effective bandwidth is around 50GB/s, which is above the STREAM bandwidth; this means the optimization techniques we employ here have greatly improve the utilization ratio of the cache.

Table 2 shows the overall HPCG performance on a single Intel Sandy Bridge CPU. Compared to the reference, the speedup is about 6.5X. For the three cases of different data size, we achieves 8.09%, 5.10%, 4.63% of the peak performance

Table 2. The overall HPCG performance on a single Intel Sandy Bridge CPU (8 cores)

Unit: GFlops	32×32×32	64×64×64	128×128×128
HPCG reference	2.18	1.33	1.25
Optimized with overhead	13.61	8.67	7.89
Optimized without overhead	13.99	8.81	8.00

without counting into the overhead introduced by transforming the sparse matrices. When the overhead is counted, the performance is around 7.88%, 5.01%, and 4.57%, for the three cases respectively.

4.2 Large-Scale Tests on Tianhe-2

We carry out large-scale tests on the Tianhe-2 supercomputer. In the tests we assign one MPI process to each CPU (12 cores), which means we initiate two MPI processes per node. The data size for each MPI process is fixed to 128×192×192.

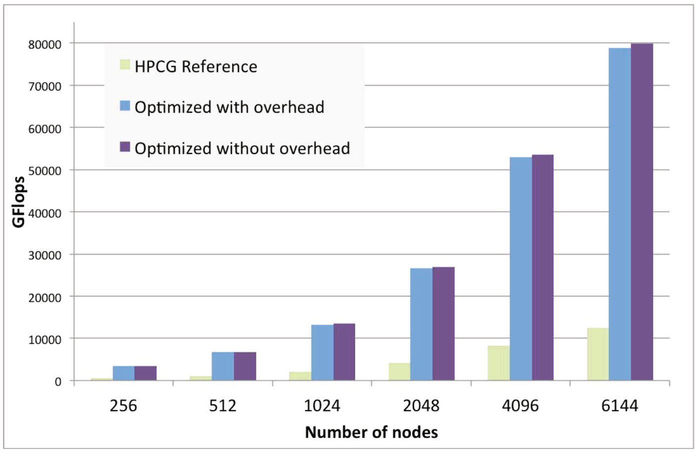


Fig. 7. Aggregate performance of HPCG on Tianhe-2

Performance results from 256 node to 6,144 nodes are presented in Fig. 7. At the largest node count, the aggregate performance is about 79.83TFlops without counting the overhead, which is 6.38X of the reference performance. When counting the overhead, the performance is only slightly decreased to about 78.82TFlops. The parallel speedup from 256 to 6,144 nodes (147,456 CPU cores) is around 23.27X, leading to a nearly ideal parallel efficiency of about 96.96%.

5 Related Works

MG, SpMV and SymGS are all important kernels that have drawn many attentions to. For optimizing these bandwidth-limited kernels, we mention works such as [9,12,13,1,10]. By carefully comparing, selecting and adjusting optimization techniques from the above works, we have optimized the related kernels in HPCG to comparable performance (in terms of both the Flops and the bandwidth usage ratio) on both an Intel Sandy Bridge computer and Tianhe-2.

There are some preliminary results on optimizing HPCG on some other platforms. Kumahata et. al [6] have conducted research on HPCG on the K computer. For the single-CPU case, they have optimized both SpMV and SymGS kernels and have achieved 2.0% of the peak performance. For multi-node, they have only tested the reference implementation. Jongsoo and Mikhail have presented their work on HPCG optimization on Intel CPUs [8]. In their work, they have obtained 2.39% of the peak performance on an Intel 12-core processor. Compared to the above works, we have optimized HPCG to achieve both a higher single-CPU performance and a much better large-scale performance on the leadership computer, Tianhe-2.

6 Concluding Remarks

In this work, we have made a first attempt to optimize and scale HPCG on the Tianhe-2 supercomputer. A variety of optimization techniques including vectorization, loop unrolling, forward-backward sweep fusion, division optimization, and OpenMP parallization have been exploited to enhance the per-socket performance. On a single Intel Sandy Bridge CPU, our optimized HPCG has outperformed the reference implementation by about 6.5X, and have sustained as high as 8.09% of the peak performance.

To reduce the cost of global communication, we have implemented a pipelined CG algorithm and properly fused some of the vector operations to reduce the addition overhead. On the Tianhe-2 supercomputer, our optimized code has scaled from 256 to 6,144 nodes (147,456 CPU cores) with a nearly ideal weak-scaling efficiency of 96.96%. The aggregated performance our code sustained on Tianhe-2 is 79.83Tflops, which is 6.38X higher than the reference implementation.

In the future, we plan to extend our research to the hybrid CPU-MIC case in order to make better usage of the computing power of Tianhe-2.

Acknowledgment. We thank NUDT and NSCC-Guangzhou for providing us free access to the Tianhe-2 supercomputer. The work was partially supported by NSF China (grants 61170075, 91130023, and 61120106005), the National Basic Research Program of China (grant 2011CB309701), and the National High-tech R&D Program of China (grant 2012AA010903).

References

1. Datta, K., Murphy, M., Volkov, V., Williams, S., Carter, J., Oliker, L., Patterson, D., Shalf, J., Yelick, K.: Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In: Proc. ACM/IEEE Conference on Supercomputing (SC 2008), pp. 4:1–4:12. IEEE Press (2008)

2. Dongarra, J., Heroux, M.A.: Toward a new metric for ranking high performance computing systems. Sandia Report SAND2013-4744, Sandia National Laboratories (2013)
3. Dongarra, J., Luszczek, P.: HPCG technical specification. Sandia Report SAND2013-8752, Sandia National Laboratories (2013)
4. García, C., Lario, R., Prieto, M., Piñuel, L., Tirado, F.: Vectorization of multigrid codes using SIMD ISA extensions. In: Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS 2003), p. 8. IEEE (2003)
5. Ghysels, P., Vanroose, W.: Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. Parallel Computing (2013) (in press)
6. Kumahata, K., Minami, K., Maruyama, N.: HPCG on the K computer. In: ASCR HPCG Workshop (2014)
7. Monakov, A., Lokhmotov, A., Avetisyan, A.: Automatically tuning sparse matrix-vector multiplication for GPU architectures. In: Patt, Y.N., Foglia, P., Duesterwald, E., Faraboschi, P., Martorell, X. (eds.) HiPEAC 2010. LNCS, vol. 5952, pp. 111–125. Springer, Heidelberg (2010)
8. Park, J., Smelyanskiy, M.: Optimizing Gauss–Seidel smoother in HPCG. In: ASCR HPCG Workshop (2014)
9. Vuduc, R., Demmel, J.W., Yelick, K.A.: OSKI: A library of automatically tuned sparse matrix kernels 16(1), 521 (2005)
10. Wellein, G., Hager, G., Zeiser, T., Wittmann, M., Fehske, H.: Efficient temporal blocking for stencil computations by multicore-aware wavefront parallelization. In: Proc. Computer Software and Applications Conference (COMPSAC 2009), vol. 1, pp. 579–586. IEEE (2009)
11. Williams, S., Kalamkar, D.D., Singh, A., Deshpande, A.M., Van Straalen, B., Smelyanskiy, M., Almgren, A., Dubey, P., Shalf, J., Oliker, L.: Optimization of geometric multigrid for emerging multi-and manycore processors. In: Proc. Int'l Conf. on High Performance Computing, Networking, Storage and Analysis (SC 2012), pp. 96:1–96:11. IEEE Computer Society Press, Los Alamitos (2012)
12. Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., Demmel, J.: Optimization of sparse matrix–vector multiplication on emerging multicore platforms. Parallel Computing 35(3), 178–194 (2009)
13. Wonnacott, D.: Using time skewing to eliminate idle time due to memory bandwidth and network limitations. In: Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS 2000), pp. 171–180. IEEE (2000)

Understanding the SIMD Efficiency of Graph Traversal on GPU[*]

Yichao Cheng, Hong An, Zhitao Chen, Feng Li, Zhaohui Wang, Xia Jiang, and Yi Peng

University of Science and Technology of China, Hefei, China
`han@ustc.edu.cn`,
`{yichao,czthack,fli186,wzhustc,jiangxia,peng1990}@mail.ustc.edu.cn`

Abstract. Graph is a widely used data structure and graph algorithms, such as breadth-first search (BFS), are regarded as key components in a great number of applications. Recent studies have attempted to accelerate graph algorithms on highly parallel graphics processing unit (GPU). Although many graph algorithms based on large graphs exhibit abundant parallelism, their performance on GPU still faces formidable challenges, one of which is to map the irregular computation onto GPU's vectorized execution model.

In this paper, we investigate the link between graph topology and performance of BFS on GPU. We introduce a novel model to analyze the components of SIMD underutilization. We show that SIMD lanes are wasted either due to the workload imbalance between tasks, or to the heterogeneity of each task. We also develop corresponding metrics to quantify the SIMD efficiency for BFS on GPU. Finally, we demonstrate the applicability of the metrics by using them to profile the performance for different mapping strategies.

Keywords: BFS, GPU, irregular computation, graph topology, SIMD underutilization, SIMD efficiency.

1 Introduction

Many practical applications (e.g., electronic design automation, geographical information, social networking and intelligent transportation systems) need to explore large-scale data sets which are represented by graphs. Recent research [13,18,19] has shown that there is abundant runtime parallelism in graph applications.

Graphics processing unit (GPU) has recently become a popular parallel platform for general computing due to its massive computational power and relatively low costs. Although many GPU-implemented applications have been

[*] We thank Xiaoqiang Li, Haibo Zhang and Tao Wang for their constructive feedback. This work is supported financially by the National Hi-tech Research and Development Program of China under contract 2012AA010902, the National Basic Research Program of China under contract 2011CB302501.

claimed to achieve more than 100x speedup, GPUs appear poorly suited for sparse graph and other irregular computation [8]. One of the reasons is that modern GPU relies on high SIMD lanes occupancy to boost performance. A full efficiency will be achieved only if all the SIMD lanes agree on their execution paths. However real-world graph instance has an irregular out-degree distribution so that the computation tasks in graph applications are intrinsically imbalanced.

Prior work [15] has attempted to assign each task with a virtual warp of threads to alleviate workload imbalance. Their results suggest that the performance is not proportional to the virtual warp size because the warp-centric method suffers from underutilization in SIMD operations, especially when the average vertex degree is smaller than the warp size. Therefore, the optimal virtual warp size is dependent on input graph and must be tuned by hand. Although their method successfully accelerates irregular graph algorithms, many questions remain unanswered. For example, what is the relationship between graph topology and SIMD efficiency? How can we quantify the SIMD efficiency for graph algorithms on GPU? What are the trade-offs among different mapping strategies? Given an input graph, how can we maximize the parallelism and SIMD efficiency?

In this work, we intend to understand the relation between graph topology, SIMD utilization and performance through parallelizing one fundamental graph algorithm on GPU: breadth-first search (BFS). BFS is a common primitive for building complex graph algorithms. In summary, this paper makes the following contributions:

– We conduct an architecture-independent characterization of graph irregularity and potential concurrency within BFS.
– We develop a model to analyze the components of SIMD underutilization. This analytical model can help us understand the relationship between graph topology and SIMD utilization. We discover that the SIMD underutilization either comes from the imbalance of vertex degree distribution, or from the heterogeneity of each vertex degree.
– We quantify the SIMD efficiency for different workload allocation strategies with three metrics derived from our model. We perform experiments to reveal the impact of SIMD efficiency on the performance of BFS on GPU. We show how to use these metrics to analyze the performance for different mapping strategies.

2 Background

2.1 GPU Architecture and CUDA Programming Model

NVIDIA GPUs consist of a scalable array of Streaming Multiprocessors (SM). Each SM contains a set of cores, called Stream Processors (SP), or CUDA cores. Multiple SPs are grouped to share a single instruction unit, and execute instructions in an SIMD (Single Instruction, Multiple Data) manner. Logically, a *warp* is a group of (e.g. 32) threads executing one common instruction at a time. Although modern GPU architectures allow threads in a *warp* to execute different instruction paths, this can severely degrade performance because the execution needs to be serialized. Threads from a warp falling into different instruction paths are said to *diverge*; Data-dependent branch is the common trigger of *divergence*. Moreover, different warps can execute actively on the same group of cores in an SMT (Simultaneous Multithreading) fashion, as a way to improve throughput despite high latencies.

CUDA is a programming model for leveraging the NVIDIA GPU for general-purpose computation. A CUDA program consists of different phases that are running either on host CPU or GPU. The GPU code is called *kernel*. A kernel is organized as a hierarchy of threads: threads are grouped into *blocks*, and blocks are grouped into a *grid*. In particular, each thread is mapped onto a CUDA core and each block is mapped onto a SM. Threads within a block can cooperate with each other through a scratchpad memory and barriers.

This abstraction hides the details of thread execution. It requires programmers to specify the hierarchical structure of threads before launching the kernel, then the underlying hardware is in charge of scheduling the threads. This scheme can improve the programming productivity but does not naturally fit the graph algorithms for two reasons. First, the intrinsic irregularity of real-world graph instance yields significant workload imbalance. So an implementation unaware of the execution model can cause thread divergence and underutilize hardware. Second, the parallelism pattern within graph algorithms depends upon the structure of input graph, which is hard to predict.

2.2 Breadth-First Search

Breadth-first search (BFS) is a fundamental graph algorithm. Given a graph $G(V, E)$, and a source vertex r, the BFS algorithm systematically traverses the edges of G and produces a BFS tree (Fig. 1) with the root r and all reachable vertices. Each node in the BFS tree is assigned with a *level* value (or minimum hops) that equals to its height in the tree. The algorithm guarantees all vertices at level k are visited before any vertices at level $k + 1$ are discovered. As shown in Fig. 1, the vertices of the same *level* compose a *vertex-frontier* and all their out-going edges logically form an *edge-frontier*. In essence, the process of BFS can be viewed as iterations over two vertex-frontiers: in each iteration, vertices in the *current* frontier are being visited, and all their unvisited neighbors are added to the *next* frontier. The process repeats until the *current* frontier is empty.

Algorithm 1 describes the kernel used in a traditional GPU implementation. As can be seen, each vertex (indicated by u) in the current frontier (C) is assigned to a single thread (Line 2). The for-loop in Line 3-8 iterates over u's neighbors and attempts to visit them. The visited neighbors are put into the next frontier (N). Noticeably, the number of iterations each thread needs to perform depends on the size of neighborhood. As will be characterized in Section 3.3, the real-world graph instances display tremendous heterogeneity. Therefore one thread

Algorithm 1. *BFS_kernel*

```
1. $tid$ = getThreadId
2. $u = C_{tid}$
3. for $v \in u$' s neighbors do
4.     if $v$ has not been visited then
5.         Visit $v$
6.         $N = N \cup \{v\}$
7.     end if
8. end for
```

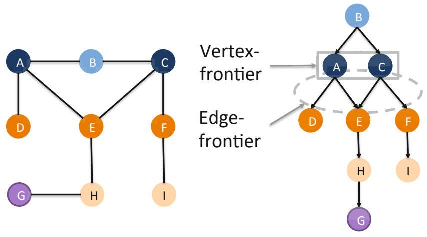


Fig. 1. BFS tree of a sample graph

having more neighbors to examine can stall the other threads within its warp, thus degrading performance.

2.3 Coarse- and Fine-Grained Mapping

Generally, there are two simple workload assignment schemes: (a) *coarse-grained mapping* assigns each thread with a single element in the vertex-frontier. Then the thread that has more neighbors to process can stall other threads within its warp; (b) *Fine-grained mapping* enlists a warp of threads to process one element in the frontier. In this case, SIMD lanes are wasted when the size of edge list of one node is smaller than the warp size.

Intuitively, the fine-grained mapping is preferable to the coarse-grained mapping for three reasons. First, the coarse-grained mapping is much more vulnerable to the irregularity of workload. Second, the fine-grained mapping exploits more concurrency, thus having a greater chance of saturating the hardware resources. Finally, the memory access pattern in the fine-grained mapping is more coalesced since the edge list of a node is accessed by a warp of threads. However the fine-gained mapping can yield higher underutilization when the average out-degree is low.

3 Dataset Characterization

3.1 Overview of Datasets

In this paper, we consider fifteen graphs from different application domains: the Florida road network is from the 9th DIMACS Implementation Challenge [2]; The Amazon co-purchase network and the LiveJournal social network are from the Stanford Large Data Collection [3]; We generate the 2D mesh datasets ourselves; The *rmat* and the *random* graph are constructed using SNAP graph library [4]; The remaining datasets are from the 10th DIMACS Implementation Challenge [1].

Table 1 characterizes these datasets in terms of scale. As can be seen, the graphs are in various sizes: the largest *random1* graph (with about 2.0 M nodes

Table 1. Graph datasets

Name	Description	Nodes	Edges	$\bar{d}$
mesh	6-point 2D mesh	1.0 M	6.3 M	6.0
FLA-road	Florida road map	1.1 M	2.7 M	2.5
thermal	3D nonlinear thermal problem	1.2 M	7.4 M	6.0
ecology	Animal/gene flow	1.0 M	4.0 M	4.0
audikw	Automotive finite element analysis	0.9 M	76.7 M	81.3
coPapers	Citation networks	0.4 M	32.0 M	73.9
LiveJournal	Social network	4.8 M	69.0 M	14.2
kkt-power	Optimal power flow	2.1 M	13.0 M	6.3
Amazon	Co-purchased products network	0.4 M	3.2 M	8.0
rmat1	Small world graph (RMAT)	5.0 M	60.0 M	12.0
rmat2	Small world graph (RMAT)	2.0 M	100.0 M	50.0
random1	Uniformly random graph	2.0 M	128.0 M	64.0
random2	Uniformly random graph	2.0 M	100.0 M	50.0
random3	Uniformly random graph	5.0 M	60.0 M	12.0
kron2	Kronecker (Graph500)	1.0 M	89.2 M	85.1

and 128.0 M edges) and the smallest *FLA-road* graph (with about 1.1 M nodes and 2.7 M edges) differ by almost two orders of magnitude. In terms of average degree ($\bar{d}$) , some graphs (e.g. *FLA-road*, *thermal*, *mesh*, *ecology*, and *kkt-power*) are sparse while others (*rmat1,2* and *random1-3*) are relatively dense. The *coPapers* and *audikw* datasets have a small node set (about 0.4 M and 0.9 M, respectively) but a considerable edge number (about 73.9 M and 81.3 M, respectively).

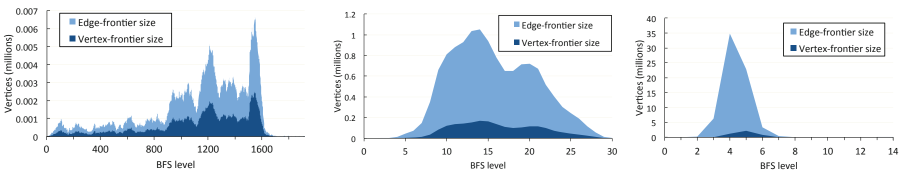
3.2 Profiling Available Concurrency During BFS

There are two levels of concurrency lying in each iteration of BFS: (a) one is among the vertices that can be processed simultaneously and (b) the other is among the outgoing edges of each vertex. So the potential concurrency during BFS can be represented by the *vertex-frontier size* and the *edge-frontier size.*

Fig. 2 plots such two-level concurrency exposed in each BFS level. As can be seen, parallelism grows slowly in the *FLA-road.* This is because most nodes in the graph have 2 or 3 neighbors. As a result, the number of vertices that can be put into the frontier is limited during each BFS iteration. Three other sparse graphs (*mesh*, *ecology* and *thermal*) also have this property. We also observed that these graphs have a much longer diameter (greater than 1000). On the contrary, the diameter of *LiveJournal* is small and most of its concurrency is lying in a few levels. This property, so-called *small world phenonmenon* [26], has also been observed in *kron20*, *rmat1,2* and *coPapers.*

3.3 Characterization of Irregularity

Although BFS exhibits abundant runtime concurrency, it is non-trivial to map the computation tasks to GPU's massive parallel resources because the



(a) *FLA-road* (b) *kkt-power* (c) *LiveJournal*

Fig. 2. Plots of vertex- and edge-frontier size in each BFS level on three sample graphs

workload is usually imbalanced. Fig. 3 depicts the out-degree distribution across the datasets in boxplots[1].

As can be seen, the leftmost four graphs (*ecology*, *mesh*, *thermal* and *FLA-road*) are observed within a tight range of out-degree: the difference between the highest and the lowest out-degree is no more than 8; and most vertices differ by only 1 out-degree. The *Amazon* graph is relatively regular, more than a half of the vertices have 10 outgoing edges, but the remaining nodes have an out-degree between 0 to 9. The small-world networks (*rmat1,2*) are pretty concentrated: most vertices are within a small range (10∼14 and 47∼53, respectively). On the other hand, the *random1-3* networks are far more irregular: vertices with different out-degree are distributed evenly between a wide range. The rightmost four graphs (*coPapers*, *kkt-power*, *LiveJournal* and *kron20*) are highly skewed. As a result, outliers exist in these graphs (up to 131,503 degree in *kron20*). Among all the datasets, the *kron20* network is the most irregular graph, which has an out-degree span of nearly 0.1 M.

4 Topology and Utilization

In this section, we introduce a model which can help us analyze the impact of graph topology on SIMD utilization. As shown in Fig. 4, we divide a warp of threads into groups and assign each group with one vertex in the frontier. Given a frontier of N vertices, we enlist N groups of threads to process them independently. Each thread group performs its work in a narrower logical SIMD window iteratively. For example, thread group 1 requires two iterations to fulfill its job and thread group 2 requires four. The logical SIMD window of group 1 can not be released due to the physical SIMD convention. The warp size is 32. The group size is denoted as S ($S \in \{1, 2, \cdots, 32\}$). The coarse-grained and the fine-grained mapping can be viewed as two special cases where $S = 1$ and $S = 32$, respectively. The work amount (out-degree) assigned to each group is denoted as d_k ($0 \le k \le N - 1$).

[1] In each boxplot, the two horizontal lines (topmost and bottommost) represent the greatest and the least value (out-degree). The top and the bottom of the box denote the first (25%) and the third *quartile* (75%), respectively. Finally, the diagonal cross in the box stands for the *median* (50%).

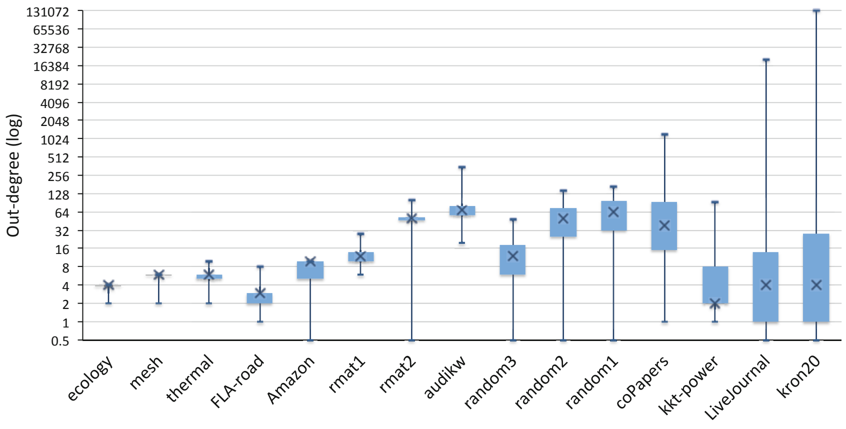


Fig. 3. Boxplot of out-degree distribution across the datasets

We divide the SIMD underutilization of each group into two parts: (a) **Intra-group underutilization** U_A^k: represented by the "bubbles" (shown here as the circles with red diagonal lines) within the logical SIMD window of group k; (b) **Inter-group underutilization** U_R^k: defined as the workload gap (represented by the dashed circles) between group k and the group having the heaviest workload in the same warp.

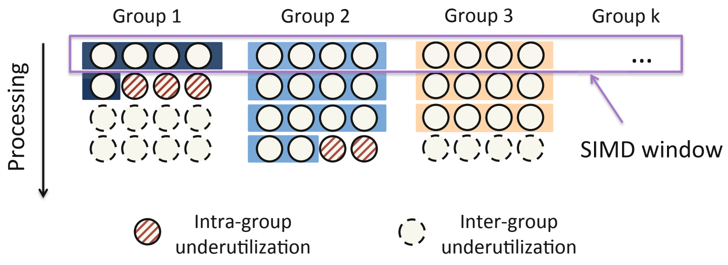


Fig. 4. Intra- and inter-group underutilization within a warp

U_A^k can be given by: $U_A^k = S \cdot \lceil \frac{d_k}{S} \rceil - d_k$. Replace $\lceil \frac{d_k}{S} \rceil$ with $\frac{d_k}{S} + \phi_k$, where $\phi_k \in [0, 1)$. So the overall intra-group underutilization can be represented by:

$$U_A = \sum_{k=0}^{N-1} S \cdot \phi_k \tag{1}$$

Then we have $U_A \in [0, N \cdot S)$. The lower bound of U_A is achieved when $S = 1$. Let D_k denote the dominating workload within the warp where group k is located. Then U_R^k is given by:

$$U_R^k = S \cdot \left(\left\lceil \frac{D_k}{S} \right\rceil - \left\lceil \frac{d_k}{S} \right\rceil \right) \tag{2}$$

Replace $\lceil \frac{D_k}{S} \rceil$with $\frac{D_k}{S} + \theta_k$, where $\theta_k \in [0, 1)$. Similarly, let $\lceil \frac{d_k}{S} \rceil = \frac{d_k}{S} + \sigma_k$, where $\sigma_k \in [0, 1)$:

$$\begin{aligned} U_R^k &= S \cdot \left[\left(\frac{D_k}{S} + \theta_k \right) - \left(\frac{d_k}{S} + \sigma_k \right) \right] \\ &= D_k - d_k + S \cdot (\theta_k - \sigma_k) \\ &= D_k - d_k + S \cdot \pi_k \quad (-1 < \pi_k < 1) \end{aligned}$$

By comparing the above equation with Eq. 2, we notice that the role of $S \cdot \pi_k$ is to round $D_k - d_k$ to the closest multiple of S. This term can be neglected if S is much smaller than $D_k - d_k$. In this case, the overall inter-group underutilization U_R can be represented by:

$$U_R = \sum_{i=0}^{N-1} (D_k - d_k) \tag{3}$$

According to Eq. 1 and Eq. 3, we can draw the following conclusions:

– U_R is primarily induced by the heterogeneity of workloads. The term $D_k - d_k$ in Eq. 3 reflects the degree of out-degree dispersion between vertices. When a graph is regular, $D_k - d_k$ is almost zero, then U_R is negligible. On the other hand, U_R is significant when traversing an irregular graph.
– U_R is sensitive to the group size. More specifically, D_k may rise as we decrease S. This is because the more groups located in a warp, the more likely that the warp will encounter a "outlier". When $S = 32$, there is only one group in the warp, so U_R is 0.
– U_A is determined by the intrinsic irregularity of vertex degree (indicated by ϕ_k). For example, if a vertex has a degree of 23, then $U_A^k = 9, 9, 1, 1, 1, 0$, when $S = 32, 16, 8, 4, 2, 1$, respectively. In practice, U_A can be effectively limited if we shrink the group size. When $S = 1$, U_A is eliminated entirely.

Given a vertex frontier, the underutilization of SIMD lanes can convert between U_R and U_A when different group sizes are chosen. When S is small, most of the SIMD lanes are wasted in the form of inter-group underutilization; conversely, when S is large, the intra-group underutilization is the dominant factor. We observe that the inter-group underutilization is greatly affected by the topology of graph.

5 Results and Discussion

In this section, we derive metrics from our model for profiling SIMD efficiency. We show that the performance of BFS on GPU is influenced by available concurrency, spatial locality, and SIMD utilization. Finally, we discuss other use scenarios where the model can be applied to.

5.1 Methodology

We adopt a two-phase algorithm similar to previous works [12, 21]. Specifically, the algorithm expands edges from a queue, collects expanded vertices into a bitmask, and compacts the bitmask to a queue before the next iteration starts. In expansion-phase, we assign each element in the queue with a variable-sized (1, 2, ..., 32) group of threads. The corresponding mapping strategies are simply referred as *sm-{1,2,...,32}*.

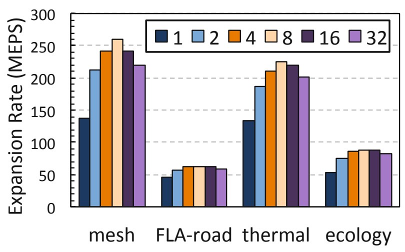
We store the graphs mentioned in Section 3.1 in Compressed Sparse row (CSR) format, which is adopted by most previous works [10, 15, 16, 20, 21, 24] for its efficiency. Our experiments are conducted using a 1.15GHz NIVIDIA Telsa C2050 GPU (Fermi Architecture) with 14 32-core SMs and 3GB of device memory. The host machine has an Intel Xeon E5506 2.13 GHz processor with 12 GB RAM. All of our programs are compiled with nvcc 5.5 and the '–O3 –arch=sm_20' flag.

In this work, we concentrate on the workload imbalance problem in expansion-phase. To benchmark the performance, we calculate the *expansion rate* in Millions of Edges Per Second (MEPS) by taking the ratio of the expanded edges to the execution time for expansion-phase. For each plotted data point, we perform BFS 10 times from 10 pseudo randomly selected vertices and average the runtime.

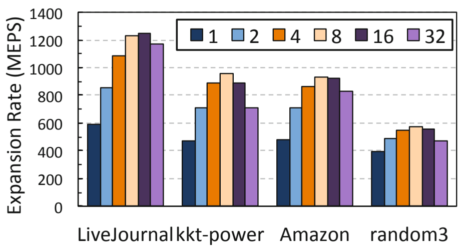
5.2 Comparing Mapping Strategies

We first compare the performance for different mapping strategies across all the datasets (Fig. 5). All the datasets fall into four categories according to their behavior:

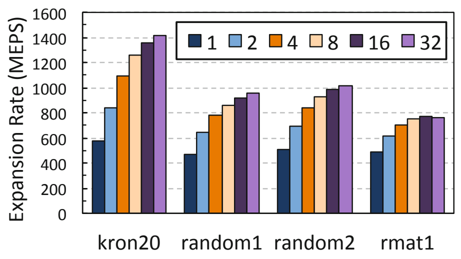
– *Low expansion rate, poor scalability.* Four datasets in this category (*mesh*, *FLA-road*, *thermal*, and *ecology*) appear poorly suited for GPU implementation. Recall that concurrency grows slowly in those graphs (Section 3.2). Therefore, the GPU implementation can not spawn sufficient amount of work to saturate the hardware in most BFS iterations. Employing a fine-grained mapping strategy can introduce more concurrency. But it also leads to higher SIMD underutilization, since the average out-degree in those datasets is low (<8). Hence, *sm-32* and *sm-16* are even slower than *sm-8*.
– *Medium expansion rate, poor scalability.* Datasets in this category (*LiveJounral*, *kkt-power*, *Amazon*, and *random3*) have considerable expansion rate due to their rich concurrency. However their performance still does not scale with the group size. Namely, *sm-32* can not outperform all its counterparts. For example, the optimal mapping granularity for *random3* and *LiveJournal* is 8 and 16, respectively. This observation is coherent with [15].
– *Medium expansion rate, good scalability.* Four graphs in this category (*kron20*, *random1,2*, and *rmat1*), on the contrary, have better scalability. That is, the expansion rate increases with group size (except for *rmat1*). Recall that most vertices in *rmat1* have an out-degree range between 10 to14 (Fig. 3). Therefore, the optimal group size for *rmat1* is 16.



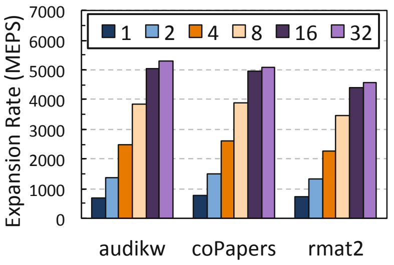
(a) Low expansion rate, poor scalability



(b) Medium expansion rate, poor scalability



(c) Medium expansion rate, good scalability



(d) High expansion rate, good scalability

Fig. 5. Expansion rate in MEPS for different mapping granularity

– *High expansion rate, good scalability.* Three datasets in this category (*audikw*, *coPapers*, and *rmat2*) not only have high expansion rate, but also scale well with group size. We observed abundant spatial locality in these datasets. Finally, the expansion rate on these datasets achieves up to ∼5000 MEPS in the case of *sm-32*.

5.3 Evaluting the SIMD Efficiency

We derive three metrics: `UR`, `UA` and `ME` from the model in Section 4 for quantifying the SIMD efficiency. `UR` and `UA` stands for the inter- and intra-group underutiliztion rate, respectivley; `ME` (*mapping efficiency*) is a metric that reflects the utilization rate for a certain workload assignment strategy. `ME` is complementary to `UA` and `ME`. We calculate these metrics online by simulating the SIMD windows operating on the vertex-frontier during BFS.

Fig. 6 captures the utilization trend with increasing group size across datasets. As can be seen, `UA` (light blue bar) always rises with the group size and `UR` (orange bar) falls with it. This is consistent with our conclusion in Section 4.

For some graphs (e.g. *kron20*, *audikw*, and *copapers*), increasing group size can effectively alleviate the inter-group underutilization with introducing minor intra-group underutilization, thus improving the mapping efficiency. This explains why these datasets have good scalability. While the SIMD utilization for some datasets (*random1,2* and *rmat2*) presents a descending trend, the mapping efficiency can be maintained at a high level (80%) when increasing group size. So the performance for these datasets scales with group size too. The mapping efficiency of *rmat1* declines by nearly half from *sm-16* to *sm-32*. This explains why the expansion rate of *sm-32* is slower than *sm-16* (Fig. 5(c)).

But for some datasets (e.g. *LiveJournal*, *kkt-power*, *random3*), the benefits of reducing inter-group undertutilization are soon outweighed by the fast-growing intra-group underutilization. For other graphs (e.g. *mesh*, *FLA-road, thermal, ecology*, and *Amazon*), increasing group size can do little help to the inter-group underutilization. On the contrary, it will only lead to severe intra-group underutilization. All these datasets correspond to the aforementioned categories that have poor scalability. The optimal group size for them is either 8 or 16.

5.4 Discussion

Our analysis shows that the performance of BFS on GPU depends on many factors. The available concurrency is the first-order impact. It is only when the input graph offers sufficient concurrency, the capabilities of GPU (massive threads, lightweight context switching, etc) can be exploited. However, when the hardware resources are saturated, the utilization of SIMD lanes becomes critical. As can be seen, different datasets favor different workload allocation granularities due to the diversity of graph structures.

Here we use the metrics to profile the SIMD efficiency. There are other scenarios where these metrics can be used: (a) *Online feedback for decision makers.* One can devise an adaptive BFS implementation that takes the SIMD efficiency into account. Notably, simulating all SIMD operations on all the vertices in every BFS level can bring huge overhead, which can be reduced by sampling (that is, by analyzing partial vertices in partial BFS levels); (b) *Feedback for auto-tuning compilers.* The optimization space of the compiler can be pruned by using these metrics to estimate the performance; (c) *Static Analysis.* One can estimate the SIMD efficiency on a target architecture by statically examining the nodes in the graph.

6 Related Work

BFS was well studied in past decades. The serial algorithm uses a queue data structure to maintain the vertex-frontier, and performs optimal $O(|V| + |E|)$ work amount [9] since each vertex/edge in the graph is visited/traversed only once. There are already massive researches on designing an efficient parallel BFS algorithm for supercomputers [6, 28] and multi-socket systems [5, 7, 20, 25, 27]. Modern GPU offers an opportunity to accelerate highly parallel graph algorithms (e.g., breadth-first search [10, 15, 16, 22, 23], connected components [14], shortest path [17]). Harish and Narayanan [11] presented a handful of CUDA

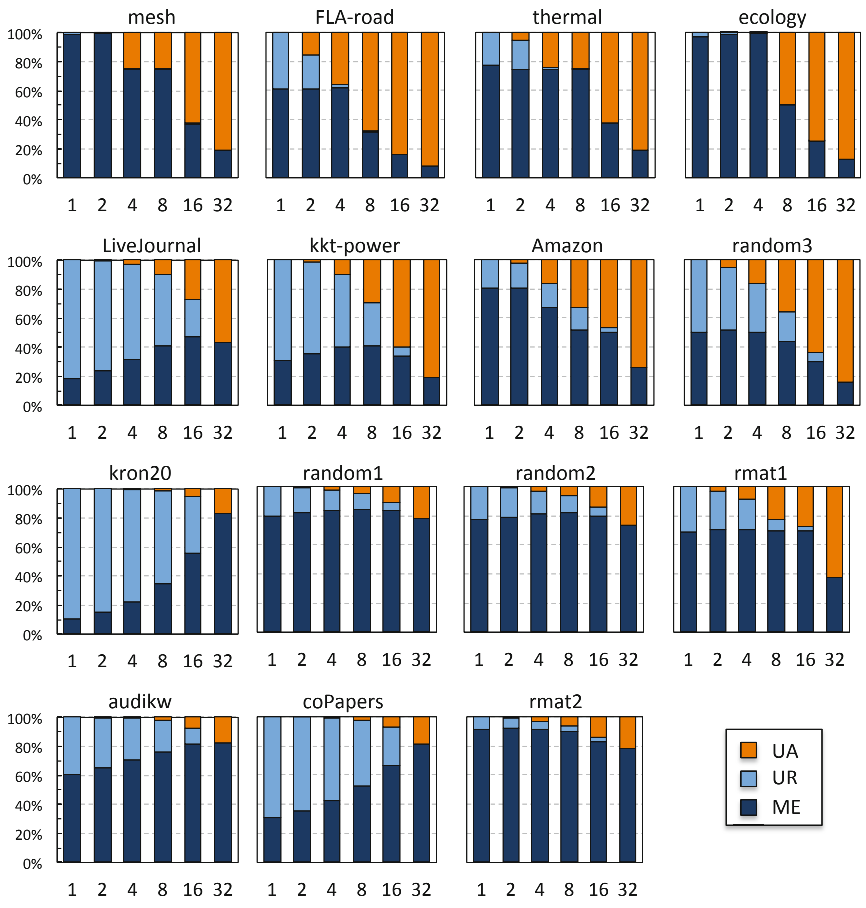


Fig. 6. SIMD utilization for different mapping strategy across datasets

implementations for graph algorithms, which is considered a pioneer research in this area. Their BFS implementation inspects all the vertices in each BFS iteration using coarse-grained mapping.

Hong et al. [15] proposed a generalized programming scheme to alleviate the load imbalance problem in graph algorithms. Their method allows trade-offs between workload imbalance and ALU underutilization by varing a single parameter (i.e., virtual warp size). Their results show that the optimal virtual warp size is dependent on the input graph and must be tuned by hand.

Merrill et al. [24] intended to achieve high SIMD utilization by leveraging prefix sums. In their implementation, threads coordinate their unique computation task by performing prefix sums to the edge lists. Therefore, no SIMD lanes are underutilized in the expansion phase. However the SIMD underutilization will occur during calculating the prefix sums. So they devise a hybrid strategy to mitigate the overhead.

Due to the heterogeneity of real-world graph datasets, some work considered an adaptive solution. Hong et al. [16] proposed a hybrid method which dynamically selects between CPU and GPU implementations for each BFS iteration. Li et al. [21] explored the implementation space for graph algorithms on GPU in different dimensions (e.g., mapping granularity, vertex-frontier representation). Their runtime uses either a thread-wise or a block-wise mapping in the expansion phase.

7 Conclusion

It is non-trivial to map the the irregular computation tasks in graph application onto GPU's hardware. Traditional simple workload allocation schemes can easily underutilize the hardware. A better solution is to assign each task with a group of threads. However, the optimal mapping granularity is usually dependent on the input graph and must be manually tuned.

In this paper, we attempt to understand the link between graph topology and SIMD utilization with one fundamental graph traversal algorithm (BFS). We present a model for analyzing the components of SIMD underutilization. Based on the model, we discover that the SIMD lanes are underutilized either due to the imbalance of vertex degree distribution, or to the heterogeneity of each vertex degree. We also devise three metrics for quantifying the SIMD efficiency. By using these metrics, the performance for different mapping strategies on a variety of datasets can be explained. We expect that the method presented in this paper will provide a foundation for developing techniques of static analysis and runtime optimization.

References

1. 10th dimacs implementation challenge, `http://www.cc.gatech.edu/dimacs10/index.shtml` (accessed: December 15, 2013)
2. 9th dimacs implementation challenge, `http://www.dis.uniroma1.it/~challenge9/download.shtml` (accessed: December 15, 2013)
3. Stanford large network dataset collection, `http://snap.stanford.edu/data/index.html` (accessed: December 15, 2013)
4. Stanford network analysis platform, `https://snap.stanford.edu/snap/index.html` (accessed: December 15, 2013)
5. Agarwal, V., Petrini, F., Pasetto, D., Bader, D.A.: Scalable graph exploration on multicore processors. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. IEEE Computer Society (2010)
6. Bader, D.A., Madduri, K.: Designing multithreaded algorithms for breadth-first search and st-connectivity on the Cray MTA-2. In: International Conference on Parallel Processing, ICPP 2006, pp. 523–530. IEEE (2006)

7. Beamer, S., Asanovic, K., Patterson, D.: Direction-optimizing breadth-first search. In: 2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1–10. IEEE (2012)
8. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., Lee, S.H., Skadron, K.: Rodinia: A benchmark suite for heterogeneous computing. In: IEEE International Symposium on Workload Characterization, IISWC 2009, pp. 44–54. IEEE (2009)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., et al.: Introduction to algorithms, vol. 2. MIT Press, Cambridge (2001)
10. Deng, Y., Wang, B.D., Mu, S.: Taming irregular EDA applications on GPUs. In: IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers, ICCAD 2009, pp. 539–546. IEEE (2009)
11. Harish, P., Narayanan, P.J.: Accelerating large graph algorithms on the GPU using CUDA. In: Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2007. LNCS, vol. 4873, pp. 197–208. Springer, Heidelberg (2007)
12. Harish, P., Vineet, V., Narayanan, P.J.: Large graph algorithms for massively multithreaded architectures. Centre for Visual Information Technology, I. Institute of Information Technology, Hyderabad, India, Tech. Rep. IIIT/TR/2009/74 (2009)
13. Hassaan, M.A., Burtscher, M., Pingali, K.: Ordered vs. unordered: A comparison of parallelism and work-efficiency in irregular algorithms. In: Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming, pp. 3–12. ACM (2011)
14. Hawick, K.A., Leist, A., Playne, D.P.: Parallel graph component labelling with gpus and cuda. Parallel Computing 36(12), 655–678 (2010)
15. Hong, S., Kim, S.K., Oguntebi, T., Olukotun, K.: Accelerating CUDA graph algorithms at maximum warp. In: Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming, pp. 267–276. ACM (2011)
16. Hong, S., Oguntebi, T., Olukotun, K.: Efficient parallel graph exploration on multicore CPU and GPU. In: 2011 International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 78–88. IEEE (2011)
17. Katz, G.J., Kider Jr., J.T.: All-pairs shortest-paths for large graphs on the GPU. In: Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware, pp. 47–55. Eurographics Association (2008)
18. Kulkarni, M., Burtscher, M., Inkulu, R., Pingali, K., Casçaval, C.: How much parallelism is there in irregular applications? ACM Sigplan Notices 44, 3–14 (2009)
19. Kulkarni, M., Pingali, K., Walter, B., Ramanarayanan, G., Bala, K., Chew, L.P.: Optimistic parallelism requires abstractions. ACM SIGPLAN Notices 42, 211–222 (2007)
20. Leiserson, C.E., Schardl, T.B.: A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers). In: Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures, pp. 303–314. ACM (2010)
21. Li, D., Becchi, M.: Deploying Graph Algorithms on GPUs: An Adaptive Solution. In: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, pp. 1013–1024 (May 2013), `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6569881`
22. Luo, L., Wong, M., Hwu, W.M.: An effective GPU implementation of breadth-first search. In: Proceedings of the 47th Design Automation Conference, pp. 52–55. ACM (2010)
23. Merrill, D., Garland, M., Grimshaw, A.: High performance and scalable gpu graph traversal. Univ. of Virginia, Tech. Rep. UVA CS-2011-05 (2011)

24. Merrill, D., Garland, M., Grimshaw, A.: Scalable GPU graph traversal. ACM SIGPLAN Notices 47, 117–128 (2012)
25. Scarpazza, D.P., Villa, O., Petrini, F.: Efficient breadth-first search on the cell/be processor. IEEE Transactions on Parallel and Distributed Systems 19(10), 1381–1395 (2008)
26. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature 393(6684), 440–442 (1998)
27. Xia, Y., Prasanna, V.K.: Topologically adaptive parallel breadth-first search on multicore processors. In: Proceedings of the 21st IASTED International Conference, vol. 668, p. 91 (2009)
28. Yoo, A., Chow, E., Henderson, K., McLendon, W., Hendrickson, B., Catalyurek, U.: A scalable distributed parallel breadth-first search algorithm on BlueGene/L. In: Proceedings of the ACM/IEEE SC 2005 Conference, Supercomputing, p. 25. IEEE (2005)

A GPU Implementation of Clipping-Free Halftoning Using the Direct Binary Search

Hiroaki Koge, Yasuaki Ito, and Koji Nakano

Department of Information Engineering, Hiroshima University,
Kagamiyama 1-4-1, Higashi Hiroshima 739-8527, Japan
{kouge,yasuaki,nakano}@cs.hiroshima-u.ac.jp

Abstract. Halftoning is an important process to convert a gray scale image into a binary image with black and white pixels. The clipping-free DBS (Direct Binary Search)-based halftoning is one of the halftoning methods that can generate high quality binary images. However, considering the computing time, it is not realistic for most applications such as printing purpose. The main contribution of this paper is to show a new GPU implementation for the clipping-free DBS-based halftoning. We have considered programming issues of the GPU architecture to implement the method on the GPU. The experimental result shows that our GPU implementation on NVIDIA GeForce GTX 780 Ti for a 4096×3072 gray scale image runs in 7.240 seconds, while the CPU implementation runs in 346.6 seconds. Thus, our GPU implementation attains a speed-up factor of 47.82.

Keywords: Image processing, Halftoning, Direct binary search, Clipping-free, GPGPU.

1 Introduction

Recent Graphics Processing Units (GPUs), which have a lot of processing units, can be used for general purpose parallel computation. Since GPUs have very high memory bandwidth, the performance of GPUs greatly depends on memory access. CUDA (Compute Unified Device Architecture) [19] is the architecture for general purpose parallel computation on GPUs. Using CUDA, we can develop parallel algorithms to be implemented in GPUs. Therefore, many studies have been devoted to implement parallel algorithms using CUDA [5,6,8,16,22,28,29].

A gray scale image is a two dimensional matrix of pixels taking a real number in the range $[0, 1]$. Usually a gray scale image has 8-bit depth, that is, each pixel takes one of the real numbers $\frac{0}{255}, \frac{1}{255}, \dots, \frac{255}{255}$, which correspond to pixel intensities. *A binary image* is also a two dimensional matrix of pixels taking a binary value 0 (black) or 1(white). *Halftoning* is an important process to convert a gray scale image into a binary image [2,13,17]. This process is necessary when a monochrome or color image is printed by a printer with limited number of ink colors.

Many halftoning techniques including Error Diffusion [7], Dot Diffusion [12], Ordered Dither using the Bayer threshold array [3] and the Void-and-Cluster threshold array [26], Direct Binary Search (DBS) [1,14], Local Exhaustive Search (LES) [10,11], have been presented.

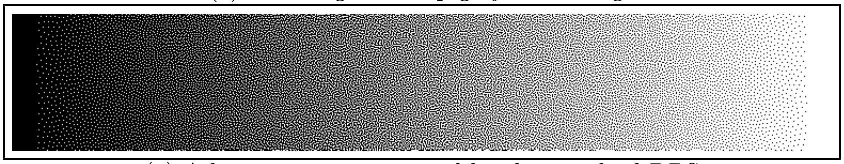
The Ordered Dither [3] uses a threshold array to generate a binary image from an original gray scale image. Each pixel of the original gray scale image is compared with an element of the threshold array. From the result of the comparison, the pixel value of the corresponding pixel of the binary image is determined. Binary images generated by the Ordered Dither method using the Bayer threshold array [3] have artifacts with periodic dots arranged in a two dimensional grid [27].

It is known that, in many cases, the DBS [1,14] generates better quality images. The key idea of the DBS is to find a binary image whose projected image onto human eyes is very close to the original image. The projected image is computed by applying a Gaussian filter, which approximates the characteristic of the human visual system. Let the total error of the binary image be the sum of the differences of the intensity levels over all pixels between the original image and the projected image. In the DBS, a pixel value is toggled if the resulting image has smaller total error. Also, neighboring pixel values are swapped if the total error of the resulting image decreases. The DBS generates a sharp binary image, especially, for middle tone areas. However, the generated binary image by the DBS has no tone in highlights and shadows. Fig. 1 and 2 show the binary images generated by the DBS. The resulting image has *clippings*, that is, highlights and shadows have no minority pixels and lose the tone of the original image. For example, several columns from the leftmost of Fig. 1 have no white pixel, although the original image has tone. Also, there is no black pixel in several columns from the rightmost and a pseudo border line appears.

For most printing devices, black pixels gain by dot-gain [11]. In other words, the average intensity level of the actual printed image is smaller than that of a binary image used for printing. If this is the case, the intensity levels of an original gray scale image are calibrated such that the actual printed image reproduces the intensity of the original gray scale image correctly. For example, suppose that intensity level 254/255 of an original gray scale image is adjusted to 1023/1024. After that, the adjusted gray scale image is converted to the binary image. Clearly, the binary image thus obtained have fewer black pixels and the average intensity is 1023/1024. However, black pixels gain by dot-gain, and the intensity level of the actual printed image will increase to 254/255. This means that, halftoning methods are required to generate the binary image with average intensity level 1023/1024. If this is not possible, actual printed images cannot reproduce intensity level 254/255, and should have tone jumps.

To avoid the clipping generated by the DBS, in [30], a DBS-based halftoning method, called *the clipping-free DBS-based halftoning* was proposed. The method is based on the DBS and can generate clipping-free binary images. The key idea of the method is to apply the Ordered Dither method using a threshold array generated by the DBS to highlights and shadows of an original gray scale image. In the method, minority pixels are preserved, that is, black pixels in the highlights and white pixels in the shadow areas, and apply DBS to the whole image.

(1) The Original ramp gray scale image


(2) A binary image generated by the standard DBS

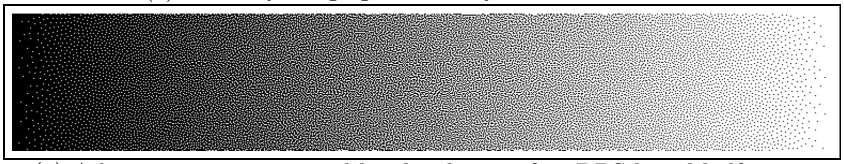

(3) A binary image generated by the clipping-free DBS-based halftoning

Fig. 1. The resulting halftone images for the ramp image

The resulting binary images have no clipping and reproduce the original tones very well. Further, the DBS-based halftoning preserves the linearity of intensity levels. In general, visually pleasing halftone textures are perceived as smooth, contain a large variety of patterns, and exhibit accurate tone rendition [15]. In other words, the resulting binary images also have high texture quality.

The resulting halftoned images generated by the DBS halftoning and the clipping-free DBS-based halftoning have high texture quality. However, compared with other well-known halftone methods, such as the Error diffusion, much more computing time is necessary. To accelerate the computation, therefore, several GPU implementations of the DBS has been proposed [4,24,25]. In [25], a GPU implementation of the DBS were proposed. Also, the implementation was extended to halftoning for color images [4] and multi-toning [24]. However, as far as we know, there is no implementation of the clipping-free DBS-based halftoning. The main contribution of this paper is to show a new GPU implementation for the clipping-free DBS-based halftoning. We have considered programming issues of the GPU architecture to implement the method. The experimental result shows that our GPU implementation on NVIDIA GeForce GTX 780 Ti for a 4096×3072 gray scale image runs in 7.240 seconds, while the CPU implementation runs in 346.6 seconds. Thus, our GPU implementation attains a speed-up factor of 47.82. Considering the computing time and the resulting clipping-free binary images, our GPU implementation is more realistic for most applications such as printing purpose.

2 Review of the Clipping-Free DBS-Based Halftoning

The main purpose of this section is to introduce the Ordered Dither [3,26] and the Direct Binary Search [1,14], and review the clipping-free DBS-based halftoning method with them as key ingredients [30].

2.1 The Ordered Dither and the Direct Binary Search

Suppose that an original gray scale image $A = (a_{i,j})$ of size $n \times n$ is given[1], where $a_{i,j}$ denotes the intensity level at position (i, j) $(0 \leq i, j \leq n-1)$ taking a real number in the range $[0, 1]$. The goal of halftoning is to find a binary image $B = (b_{i,j})$ of the same size that reproduces the original image A, where each $b_{i,j}$ is either 0(black) or 1(white). The Ordered Dither uses a threshold array $T = (t_{i,j})$ of size $m \times m$, with each element taking a real number $\frac{0}{255}, \frac{1}{255}, \ldots,$ or $\frac{254}{255}$. More specifically, the pixel value of each pixel $b_{i,j}$ is determined by the following formula:

$$b_{i,j} = \begin{cases} 0 \text{ if } a_{i,j} \leq t_{i \bmod m, j \bmod m} \\ 1 \text{ if } a_{i,j} > t_{i \bmod m, j \bmod m} \end{cases}$$

Note that since $b_{i,j}$ is always 0 if $t_{i \bmod m, j \bmod m} = \frac{255}{255}$, the threshold value never takes $\frac{255}{255}$. The Bayer halftoning uses the Bayer threshold array which is defined recursively [3].

The idea of the DBS is to measure the goodness of the output binary image B using the Gaussian filter that approximates the characteristic of the human visual system. Let $V = (v_{k,l})$ denote a Gaussian filter, i.e. a 2-dimensional symmetric matrix of size $(2w + 1) \times (2w + 1)$, where each non-negative real number $v_{k,l}$ $(-w \leq k, l \leq w)$ is determined by a 2-dimensional Gaussian distribution such that their sum is 1. In other words,

$$v_{k,l} = c \cdot e^{-\frac{k^2+l^2}{2\sigma^2}}, \tag{1}$$

where σ is a parameter of the Gaussian distribution and c is a fixed real number to satisfy $\sum_{-w \leq k,l \leq w} v_{k,l} = 1$. Let $R = (r_{i,j})$ be the projected gray scale image of a binary image $B = (b_{i,j})$ obtained by applying the Gaussian filter as follows:

$$r_{i,j} = \sum_{-w \leq k,l \leq w} v_{k,l} b_{i+k,j+l} \quad (0 \leq i, j \leq n-1). \tag{2}$$

Clearly, from $\sum_{-w \leq k,l \leq w} v_{k,l} = 1$ and $v_{k,l}$ is non-negative, each $r_{i,j}$ takes a real number in the range $[0, 1]$ and thus, the projected image R is a gray scale image. We can say that a binary image B is a good approximation of original image A if the difference between A and R is small enough. Hence, we define the error of B as follows. The error $e_{i,j}$ at each pixel location (i, j) is defined by

$$e_{i,j} = a_{i,j} - r_{i,j} \tag{3}$$

[1] For simplicity, we assume that images are square.

and the total error is defined by

$$Error(A, B) = \sum_{0 \le i,j \le n-1} |e_{i,j}|^2. \quad (4)$$

Since the Gaussian filter approximates the characteristics of the human visual system, we can think that image B reproduces original gray scale image A if $Error(A, B)$ is small enough. The best binary image that reproduces A is a binary image B which is given by the following formula:

$$B^* = \arg\min_B Error(A, B). \quad (5)$$

It is very hard to find the optimal binary image B^* for a given gray scale image A. The idea of the DBS is to find a near optimal binary image B such that $Error(A, B)$ is sufficiently small. For this purpose, the DBS repeats improvement of binary image B. The value of a particular pixel $b_{i,j}$ is modified by the following two operations:

Toggling This operation is to toggle the value of $b_{i,j}$, that is, $b_{i,j} \leftrightarrow 1 - b_{i,j}$. The value of $b_{i,j}$ is toggled if $Error(A, B)$ decreases.

Swapping Let $b_{i',j'}$ be a neighbor pixel of $b_{i,j}$, that is, both $|i - i'| \le 1$ and $|j - j'| \le 1$ hold. This operation is to exchange the values of $b_{i,j}$ and $b_{i',j'}$, that is $b_{i,j} \leftrightarrow b_{i',j'}$. Swapping operation is performed if $Error(A, B)$ decreases.

Clearly, toggling and swapping operations do not increase the error and improve the binary image B. In the DBS, these operations are executed in the raster order. Further, this raster order improvement is repeated until no more improvement by toggling or swapping operations is possible.

Although the DBS generates high-quality binary images, it does not work well in highlights and shadows. It has *clippings*, that is, the highlight and the shadow areas have no dots and lose the tone of the original image. Fig. 1 shows the resulting binary images by the DBS. The left shadow area has no white dots and the tone is lost. Similarly, in the right highlight area, black dots disappear. Fig. 2 shows the resulting binary image generated by the DBS. Black dots in the woman's face are lost and pseudo borders appear. Also, white dots in her hair are removed. In [30], the detailed explanation about the reason of clippings by the DBS is shown.

2.2 The Clipping-Free DBS-Based Halftoning

In the following, we review the clipping-free DBS-based halftoning proposed in [30]. The key idea of this DBS-based halftoning method is to use the Ordered Dither for the pixels with intensity smaller than D or larger than $1 - D$ and then use the DBS.

First a threshold array $T = (t_{i,j})$ of size $m \times m$ used for shadows is determined. Since this T is used for the pixel values no more than D, it is not necessary to determine threshold value larger than D. Thus, for each i $(0 \le i \le D)$, $\frac{m^2}{255}$

elements in T takes value $\frac{i}{255}$. For highlight pixel with intensity larger than $1 - D$, we can use a threshold array $T' = (t'_{i,j})$ such that $t'_{i,j} = 1 - t_{i,j}$.

The goal of determining T is to distribute the threshold values in T uniformly. The uniformity is defined as follows. Let $u(i, j)$ denote the Euclidean distance to a closest threshold value no more than $t_{i,j}$. In other words,

$$u(i, j) = \min\{\sqrt{(i - i')^2 + (j - j')^2} \mid t_{i',j'} \leq t_{i,j}\}.$$

The uniformity $u(T)$ of T is the sum of $u_{i,j}$, that is,

$$u(T) = \sum_{0 \leq i,j \leq m-1} u(i, j).$$

Clearly, if threshold value distributed more uniformly, the uniformity $u(T)$ is larger.

The threshold value is assigned $\frac{0}{255}$ to $\frac{m^2}{255}$ elements in T. For this purpose, we select $\frac{m^2}{255}$ elements in T at random and assign $\frac{0}{255}$ to them. After that, we move each $\frac{0}{255}$ to a neighbor element if the uniformity $u(T)$ increases. The reader should have no difficulty to confirm that, this operation is very similar to swapping operation of the DBS. This swapping operation is repeated until no more improvement on $u(T)$ is possible. Next, we assign threshold value $\frac{1}{255}$ to $\frac{m^2}{255}$ elements in T. Similarly, we select $\frac{m^2}{255}$ elements that were not assigned $\frac{0}{255}$ and assign $\frac{1}{255}$ to them. After that, the swapping operation is performed for these elements with threshold value $\frac{1}{255}$. The same procedure is repeated until threshold values from $\frac{0}{255}$ to D are determined. If T is small, the generated binary image may have periodic artifact with frequency $m \times m$ pixels. Thus, T should be as large as possible. In the experiment [30] a threshold array of size 512×512 is large enough.

We are now in position to explain the clipping-free DBS-based halftoning. Suppose that a gray scale image $A = (a_{i,j})$ to be halftoned is given. We first apply the threshold array T to pixels $a_{i,j}$ of A such that $a_{i,j} < D$ or $a_{i,j} > 1 - D$, and obtain a binary image $B = (b_{i,j})$. Next, we assign label *determined/undetermined* to every pixel as follows:

$b_{i,j}$ is *determined*, if ($a_{i,j} < D$ and $b_{i,j} = 1$) or ($a_{i,j} > 1-D$ and $b_{i,j} = 0$), and
$b_{i,j}$ is *undetermined*, otherwise.

In other words, if $b_{i,j}$ is minority pixel in shadows or in highlights, then it is a determined pixel. After that, the DBS is executed for all undetermined pixels, that is, toggling and swapping operations repeated in the raster scan order until no more improvement of the error is possible.

According to the above, the outline of the clipping-free DBS-based halftoning that computes a halftoned binary image of an original gray scale image A is as follows:

[The clipping-free DBS-based halftoning]

Step 1: Initialization
We first assign label determined/undetermined to every pixel by applying a threshold array T. We obtain a binary image $B = (b_{i,j})$ such that
if label is determined, $b_{i,j}$ is halftoned by a threshold array T, and
if label is undetermined, $b_{i,j}$ is halftoned by *the random dither method.*
In the random dither method, a binary pixel takes value 1 with probability p if the pixel value of the corresponding pixel of an original image is p $(\in [0, 1])$. Thus, $b_{i,j} = 1$ with probability $a_{i,j}$ for every i and j except determined pixels. In Step 2, determined pixels are fixed and the DBS is performed for undetermined pixels.

Step 2: DBS
We pick an undetermined pixel $b_{i,j}$ in B one by one from the top-left corner to the bottom-right corner in the raster scan order. We select one of the operations toggling and swapping, which minimizes the total error, and update pixels by such operation. This update procedure by the raster scan order is repeated until one round of raster scan search from the top-left corner to the bottom-right corner does not update pixels and the error is not improved.

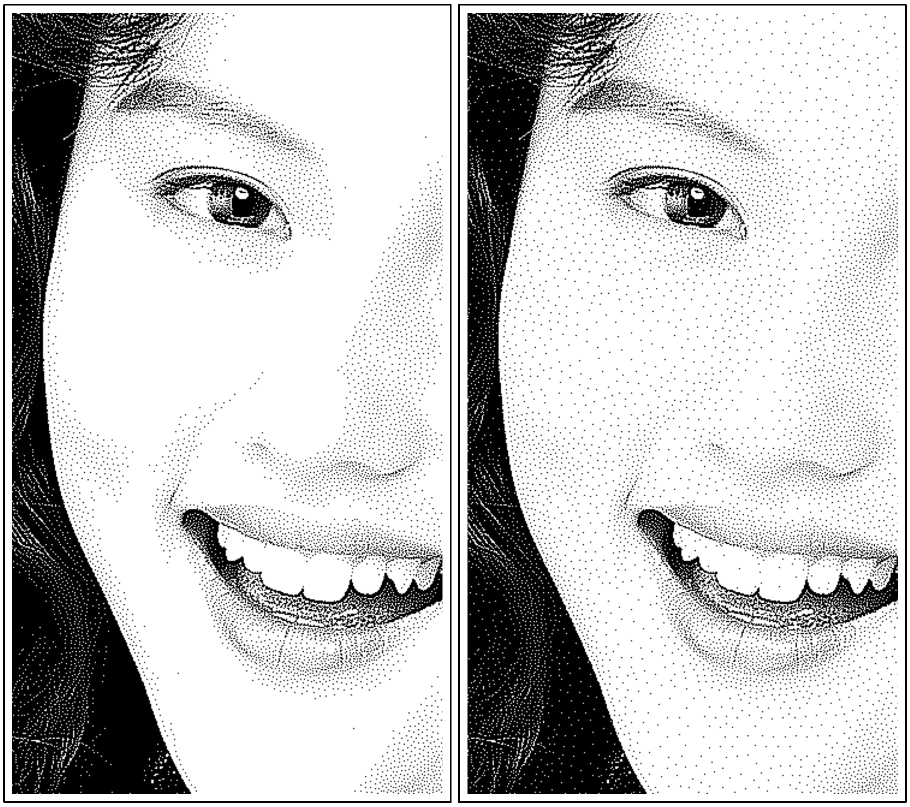
Fig. 1 and 2 show the resulting binary images. To obtain these images, we have generated a threshold array of size 512×512. We have also used the Gaussian filter with parameter $\sigma = 1.2$ and $w = 4$, and the threshold value $D = \frac{9}{255}$ is used. We can see clearly the original tone is preserved in shadows and highlights. The resulting images look smooth and contain more variety of patterns. They take on good visual texture.

3 Implementation of the Clipping-Free DBS-Based Halftoning

Before explaining our GPU implementation of the clipping-free DBS-based halftoning, in this section, we show how the clipping-free DBS-based halftoning is implemented as a sequential implementation.

In Step 1, we first assign label determined/undetermined to every pixel using a threshold array T and make a label map $L = (l_{i,j})$ of size $n \times n$ such that if $l_{i,j} = 1$ if label determined is assigned to pixel (i, j), and $l_{i,j} = 0$, otherwise. Referring L, we obtain a binary image B for determined pixels and undetermined pixels by thresholding with T and the random dither method, respectively.

In Step 2, we first compute the projected gray scale image $R = (r_{i,j})$ of the binary image B by computing formula (2). We compute the error matrix E by computing formula (3) and the total error from formula (4). In Step 2, we need to perform local search by toggling and swapping that minimize the total error. It is sufficient to compute the total error of the affected region that includes the neighboring 8 pixels. The affected region is a region of the image B such that the Gaussian filter for $b_{i,j}$ and its neighboring 8 pixels affects the pixel values of


(1) A binary image generated by the standard DBS

(2) A binary image generated by the clipping-free DBS

Fig. 2. The resulting binary images (partial enlargement) for a woman image [9]

the blurred image. More specifically, the affected region is a set $\mathcal{A}_{i,j}$ of positions in the image such that

$$\mathcal{A}_{i,j} = \{(i', j') | i - w - 1 \leq i' \leq i + w + 1, j - w - 1 \leq j' \leq j + w + 1\}.$$

Since the size of the Gaussian filter is $(2w + 1) \times (2w + 1)$, that of the affected region is $(2w + 3) \times (2w + 3)$. Therefore, in the local search by toggling and swapping, we compute the total error at pixel location (i, j) by evaluating the following formula:

$$\sum_{(i',j') \in \mathcal{A}_{i,j}} |e_{i',j'}|^2. \tag{6}$$

We evaluate this formula for each operation of toggling and swapping, and replace pixels with the minimum total error.

To perform the local search by toggling and swapping, we need to compute the convolution in formula (2) for each operation of toggling and swapping. Since pixel values of B are 0 or 1, $r_{i,j}$ can be computed by adding/subtracting the values of the Gaussian filter $v_{k,l}$ to/from $r_{i,j}$ in $\mathcal{A}_{i,j}$. For example, when a pixel $b_{i,j}$ is changed from 0 to 1 by toggling, the values of the Gaussian filter $v_{k,l}$ are only added to $r_{i,j}$ in $\mathcal{A}_{i,j}$. We note that once the total error E is computed, the update of E by toggling and swapping can be also computed by adding/subtracting the values of the Gaussian filter. It can be performed without the update of the projected image R. Therefore, in our implementation, we directly update the total error E.

To obtain further acceleration, we use an update map. In Step 2, a round of the raster scan order search is repeated. It is possible that an area of a binary image is fixed in an earlier round, and no pixels in the area are not updated until Step 2 terminates. Hence, it makes sense to perform the local search by toggling and swapping for which pixels might be updated, we use an update map $M = (m_{i,j})$ of size $n \times n$. Before a round of the raster scan order search, all values in M is initialized by 0. We set $m_{i,j} = 1$ if the operation updates pixel $b_{i,j}$, that is, the value of $b_{i,j}$ is changed from 0 to 1 or from 1 to 0. Clearly, at the end of the round, $m_{i,j} = 1$ if $b_{i,j}$ has been updated in this period. Further, the affected region in which pixels might be updated in the next round consists of (i, j) such that $m_{i,j}$ or its neighbor takes value 1.

4 GPU Implementation

The main purpose of this section is to show our GPU implementation of the clipping-free DBS-based halftoning.

We briefly explain CUDA architecture that we will use. NVIDIA provides a parallel computing architecture called *CUDA* on NVIDIA GPUs. CUDA uses two types of memories in the NVIDIA GPUs: *the global memory* and *the shared memory* [20]. The global memory is implemented as an off-chip DRAM of the GPU, and has large capacity, say, 1.5-6 Gbytes, but its access latency is very long. The shared memory is an extremely fast on-chip memory with lower capacity, say, 16-48 Kbytes. Fig. 3 illustrates the CUDA hardware architecture.

CUDA parallel programming model has a hierarchy of thread groups called *grid*, *block* and *thread*. A single grid is organized by multiple blocks, each of which has equal number of threads. The blocks are allocated to streaming multiprocessors such that all threads in a block are executed by the same streaming multiprocessor in parallel. All threads can access to the global memory. However, threads in a block can access to the shared memory of the streaming multiprocessor to which the block is allocated. Since blocks are arranged to multiple streaming multiprocessors, threads in different blocks cannot share data in the shared memories.

We are now in a position to explain how we implement the clipping-free DBS-based halftoning. We assume that an original gray scale image A of size $n \times n$ to be halftoned is stored in the global memory in advance, and the implementation

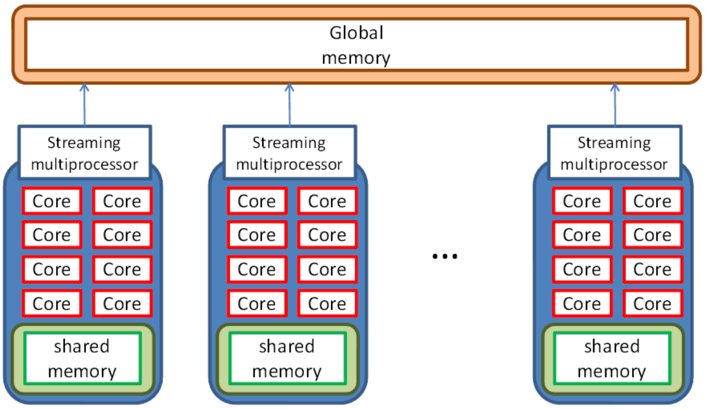


Fig. 3. CUDA hardware architecture

writes the resulting binary image B' in the global memory. Further, we assume that the threshold array T is also stored in the global memory. In the following, to perform the computation in parallel, basically we divide an input image into subimages whose size is $k \times k$ and perform halftoning for each subimage in parallel. In our implementation, the size of the subimage is $\frac{n}{k} \times \frac{n}{k}$.

To implement Step 1, $\frac{n^2}{k^2}$ CUDA blocks are invoked one for subimages of size $\frac{n}{k} \times \frac{n}{k}$. Each CUDA block is responsible for generating an initial binary image $B = (b_{i,j})$ and computing the error matrix $E = (e_{i,j})$ of the corresponding subimage using the shared memory. For this purpose, each block copies pixel values in A of their affected region $\mathcal{A}_{i,j}$ in the subimage to the shared memory. After that, threads in each block concurrently assign label determined/undetermined to every pixel and generate an initial binary image for determined/undetermined pixels by thresholding with T and the random dither method, respectively. Finally, the error matrix $E = (e_{i,j})$ of the corresponding block is computed from the blurred image of B and pixel values in A of the affected region $\mathcal{A}_{i,j}$. The error matrix E of the resulting block is copied to the global memory.

In Step 2, a kernel is invoked for each round in the DBS. In each kernel, the DBS to evaluate formula (5) is performed in parallel using multiple CUDA blocks. Each CUDA block is responsible for executing the DBS of the corresponding subimage. The local search in the DBS is performed for pixels that are assigned to label undetermined and might be updated by referring label map L and update map M.

However, the operations toggling and swapping for adjacent blocks cannot be executed in parallel, because the application of the Gaussian filter to adjacent blocks affects each other. Thus, we partition blocks into four groups such that Group 1: even columns and even rows, Group 2: odd columns and even rows, Group 3: even columns and odd rows, and Group 4: odd columns and odd rows. The reader should refer to Fig. 4 illustrating the groups. We use $\frac{4n^2}{k^2}$ CUDA

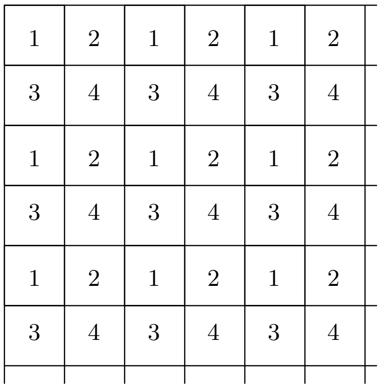


Fig. 4. Groups of blocks

blocks, and perform the local search in all blocks of each group. Note that, if $k \geq 2w$ then the Gaussian filter of two blocks in a group never affect each other, where the subimage is $k \times k$ and the size of the Gaussian filter is $(2w+1) \times (2w+1)$. In other words, the affected regions of a particular group do not overlap each other. Actually, in our experiment, we choose $k = 32$ and $w = 4$. Step 2 performs the local search for Group 1, Group 2, Group 3, and Group 4, in turn. A CUDA block is invoked for each block of a group. The CUDA block copies the error matrix corresponding to the affected region in the global memory to the shared memory.

After that, each CUDA block performs the local search by toggling and swapping for the corresponding subimage in raster scan order to obtain the best combination of pixels in B. Concretely, multiple threads in a block perform the local search pixel by pixel for the corresponding subimage in the raster scan order. In the local search, formula (6) is evaluated for each operation of toggling and swapping. In our implementation, we utilize a summing technique by binary reduction proposed in [18] to evaluate the formula.

Finally, the updated binary image and the error matrix are copied to the global memory. Some readers may think that since the local search is concurrently performed using the partition shown in Fig. 4, the total error by computing formula (4) increases compared with that by the sequential one. However, in our experiment, the total errors are almost the same and the quality of the resulting binary images cannot be distinguished.

5 Experimental Results

The main purpose of this section is to show the experimental results. We have used three gray scale images, Lena [23], Woman [9], and Flower basket [9], of size 512×512, 2048×2560, and 4096×3072, respectively. We use the Gaussian filter with parameter $\sigma = 1.2$ and $w = 4$. In the clipping-free DBS-based halftoning, we have generated a threshold array of size 512×512 and the threshold value $D = \frac{9}{255}$ is used.

In order to evaluate the computing time for generating the halftoned images, we have used NVIDIA GeForce GTX 780 Ti, which has 2880 processing cores in 15 SMX units [21]. We have also used Intel PC using Xeon X7460 running in 2.66GHz to evaluate the implementation by sequential algorithms. Table 1 shows the computing time for generating the binary images. The computing time is average of 10 times execution and the computing time of the GPU includes data transfer time between the main memory and the device memory in the GPU. Using the GPU, the computing time can be reduced by a factor of 35.88-47.82. Even if the large image is halftoned, the computing time is 7.240s by the GPU acceleration. This computing time is acceptable for most applications such as printing purpose.

Table 1. Computing time (in seconds) of the clipping-free DBS-based halftoning

Image (size)	Lena (512×512)	Woman (2048×2560)	Flower basket (4096×3072)
Intel CPU	8.351	113.3	346.6
NVIDIA GPU	0.2138	3.158	7.248
Speed-up	39.06	35.88	47.82

Additionally, according to the result of an existing GPU implementation of the original DBS, they reported that the execution time for two gray scale images of size 194×270 and 1536×1920 was 9.36s and 127s, respectively [25]. Since utilized GPUs and images differ and their implementation does not support clipping-free halftoning, it is difficult to compare their results with our results directly. Considering the computing time, however, it is clear that our GPU implementation is better than that of [25].

6 Conclusions

In this paper, we have proposed an implementation of the clipping-free DBS-based halftoning. In our implementation, we have considered programming issues of the GPU architecture. We have implemented it on NVIDIA GeForce GTX 780 Ti. The experimental result shows that our GPU implementation on NVIDIA GeForce GTX 780 Ti for a 4096×3072 gray scale image runs in 7.240 seconds, while the CPU implementation runs in 346.6 seconds. Thus, our GPU implementation attains a speed-up factor of 47.82. According to the results, we think that the computing time of our GPU implementation is realistic for most applications such as printing purpose.

References

1. Analoui, M., Allebach, J.: Model-based halftoning by direct binary search. In: Proc. SPIE/IS&T Symposium on Electronic Imaging Science and Technology, vol. 1666, pp. 96–108 (1992)
2. Asano, T., Nakano, K.: Halftoning through optimization of restored images – a new approach with hardware acceleration. Tech. rep., The Institute of Electronics Information and Communication Engineers, COMP2002-75 (March 2003)
3. Bayer, B.: An optimum method for two-level rendition of continuous-tone pictures. In: IEEE International Conference on Communications, pp. 11–15 (1973)
4. Chandu, K., Stanich, M., Trager, B., Wu, C.W.: A GPU implementation of color digital halftoning using the Direct Binary Search algorithm. In: Proc. of IEEE International Symposium on Circuits and Systems, pp. 185–188 (2012)
5. Diaz, J., Muñoz-Caro, C., Niño, A.: A survey of parallel programming models and tools in the multi and many-core era. IEEE Transactions on Parallel and Distributed Systems 23(8), 1369–1386 (2012)
6. Farivar, R., Rebolledo, D., Chan, E., Campbell, R.H.: A parallel implementation of k-means clustering on GPUs. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 340–345 (July 2008)
7. Floyd, R., Steinberg, L.: An adaptive algorithm for spatial gray scale. In: Proc. of International Symposium Digest of Technical Papers, Society for Information Displays, pp. 36–37 (1975)
8. Harish, P., Narayanan, P.J.: Accelerating large graph algorithms on the GPU using CUDA. In: Proceedings of the 14th International Conference on High Performance Computing, pp. 197–208 (2007)
9. ISO/IEC International Standard 12640: Graphic technology – prepress digital data exchange – CMYK standard colour image data, CMYK/SCID (1997)
10. Ito, Y., Nakano, K.: FM screening by the local exhaustive search with hardware acceleration. International Journal on Foundations of Computer Science 16(1), 89–104 (2005)
11. Ito, Y., Nakano, K.: A new FM screening method to generate cluster-dot binary images using the local exhaustive search with FPGA acceleration. International Journal on Foundations of Computer Science 19(6), 1373–1386 (2008)
12. Knuth, D.: Digital halftones by dot diffusion. ACM Transactions on Graphics 6(4), 245–273 (1987)
13. Lau, D.L., Arce, G.R.: Modern Digital Halftoning. Marcel Dekker (2001)
14. Lieberman, D.J., Allebach, J.P.: Efficient model based halftoning using direct binary search. In: Proc. of International Conference on Image Processing, vol. 1, pp. 775–778 (1997)
15. Lieberman, D.J., Allebach, J.P.: A dual interpretation for direct binary search and its implications for tone reproduction and texture quality. IEEE Transactions on Image Processing 9(11), 1950–1963 (2000)
16. Man, D., Uda, K., Ueyama, H., Ito, Y., Nakano, K.: Implementations of parallel computation of Euclidean distance map in multicore processors and GPUs. In: Proceedings of International Conference on Networking and Computing, pp. 120–127 (2010)
17. Nakano, K.: Various screening methods. Convertech 36(1), 72–77 (2008)
18. Nakano, K.: Optimal parallel algorithms for computing the sum, the prefix-sums, and the summed area table on the memory machine models. IEICE Transactions on Information and Systems E96-D(12), 2626–2634 (2013)

19. NVIDIA Corporation: CUDA ZONE, `http://www.nvidia.com/page/home.html`
20. NVIDIA Corporation: CUDA C Programming Guide Version 5.5 (2013)
21. NVIDIA Corporation: NVIDIA next generation CUDA compute architecture: Kepler GK110 whitepaper (2013)
22. Ogawa, K., Ito, Y., Nakano, K.: Efficient Canny edge detection using a GPU. In: International Workshop on Advances in Networking and Computing, pp. 279–280 (November 2010)
23. Po, L.M.: Lenna 97: A complete story of Lenna (2001), `http://www.ee.cityu.edu.hk/~lmpo/lenna/Lenna97.html`
24. Trager, B., Chandu, K., Wu, C.W., Stanich, M.: A GPU based implementation of Direct Multi-bit Search (DMS) screen algorithm. In: IS&T/SPIE Electronic Imaging, vol. 8655, pp. 86550Z-1–86550Z-10 (2013)
25. Trager, B., Wu, C.W., Stanich, M., Chandu, K.: GPU-enabled parallel processing for image halftoning applications. In: Proc. of IEEE International Symposium on Circuits and Systems, pp. 1528–1531 (2011)
26. Uichney, R.: The void-and-cluster method for dither array generation. In: IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology, pp. 332–343. International Society for Optics and Photonics (1993)
27. Ulichney, R.: Halftone characterization in the frequency domain. In: Proc. of IS&T's 4th Annual Conference, pp. 464–467 (1994)
28. Wang, S., Cheng, S., Wu, Q.: A parallel decoding algorithm of LDPC codes using CUDA. In: Proceedings of Asilomar Conference on Signals, Systems, and Computers, pp. 171–175 (October 2008)
29. Wei, Z., JaJa, J.: Optimization of linked list prefix computations on multithreaded GPUs using CUDA. In: Proceedings of International Parallel and Distributed Processing Symposium (2010)
30. Zhuge, X., Nakano, K.: Clipping-free halftoning and multitoning using the direct binary search. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E92-A(4), 1192–1201 (2009)

A Reliable and Secure GPU-Assisted File System

Shang-Chieh Lin, Yu-Cheng Liao and Yarsun Hsu

Department of Electrical Engineering, National Tsing Hua University, HsinChu, Taiwan, R.O.C

Abstract. This work revised the original GPU-accelerated file system to enhance not only reliability but also security. It is designed to have a flexible configuration on both reliability and encryption schemes. Different encoding levels and encryption modes can be configured for each file. Moreover, a runtime framework is proposed, which provides a unified interface for applications to easily take advantage of the various computation powers on a heterogeneous environment. Multiple devices and platforms, such as CUDA and OpenCL can be utilized at the same time to achieve a better performance.

The system is implemented with Cauchy Reed-Solomon (CRS) encoding/decoding operations for reliability and Advanced Encryption Standard (AES) encryptions for security. Both of the operations are accelerated by CUDA and OpenCL.

Finally, The runtime framework and the system were evaluated and compared with different hardware environments. The results show that the system runs efficiently and has a performance gain up to 104.57x on AES operations.

Keywords: GPGPU, CUDA, OpenCL, Filesystem, AES.

1 Introduction

In recent years, the GPGPU computing has become a popular solution for high performance computing. Various frameworks and platforms have been proposed to leverage the computation power of GPUs. However, the integration of different frameworks and the utilization of multiple computation devices become challenging for programmers. There are often big efforts to parallelize and distribute the tasks across the devices and platforms efficiently. With more and more computation resources available, the integrations and the optimizations over the system become more complex.

Previous work [1] proposed a GPU-accelerated file system prototype-*CRSFS*, which enhances the reliability by using CRS(Cauchy Reed-Solomon) erasure code [2]. It adopts the CUDA platform to accelerate the CRS encoding/decoding procedures. Unlike other hardware implementations, such as RAID, the software-based system provides a more flexible environment for users to configure the reliability settings for each file, yet the performance is still great for the acceleration by GPUs.

The objective of this work is to propose a new runtime framework on heterogeneous computing, providing an infrastructure and basic functions for programmers to implement and optimize their system efficiently. The framework runs as a middle-ware between application and other computing frameworks, and is designed to have a unified interface for task managements across computation devices. The common GPU computation flow can be easily pipelined and optimized to increase the overall throughput.

Another objective of this work is the revision of CRSFS file system prototype. The newly developed framework is integrated into the file system and the file system is added with security configurations. The encryption procedures are also accelerated by GPUs and integrated into the framework and the file operations are redesigned and modified to cope with the new encryption routines.

2 Related Work

Many runtimes and frameworks have been developed to provide a higher level interface. Sine then, programmers can focus on algorithmic concerns, rather than handling low-level issues. For instances, the CrystalGPU [3], adopted by the original CRSFS, is a framework designed for CUDA to utilize the GPU power transparently and efficiently. For another example, StarPU [4] is a C programing library for GPGPU, and is designed to be a unified runtime system for heterogeneous multicore architectures. It handles various runtime concerns, including task dependencies, heterogeneous scheduling, data transfers and so on. On the other hand, Aparapi [5] is a higher level API for expressing data parallel workload in Java, which extends the Java promise of 'Write Once Run Anywhere' to include GPU.

As for encryptions and file systems, eCryptfs [6] is a package of disk encryption software for Linux. It provides filesystem-level encryption, which is allowed to selectively apply encryption on a file or a directory. For GPU accelerated file system, GPUstore [7], presented by Weibin Sun, is a framework for integrating GPU computing into storage system. It is built on top of the CUDA framework and is integrated into the Linux Kernel.

In addition, some researches have implemented the AES algorithm with GPU accelerations. For example, Deguang Le [8] designed a parallel implementation for AES, which achieved up to 7x speedup over the implementation on CPU. Keisuke Iwai and Takakazu Kurokawa [9] gave great details on the implementation and analysis with CUDA. All of these works above show the great potential of programming on heterogeneous platform and GPGPU programming.

3 Design and Implementation

In this section, the original CRSFS in previous work is reintroduced with new features and some modifications. There are three major components in this system , which are the file system, the GPU runtime, and the kernel implementations. Details are given below respectively.

3.1 File System and FUSE

The file system is implemented by FUSE and run as a stackable file system, which does not store data itself. Instead, it uses another file system for storage. The encoding/decoding and encryption procedures are inserted in the FUSE framework and will be triggered automatically by the user's file operations. The design is to be flexible that it can be easily applied to any other existing file system and is able to configure the coding schemes for each file.

The system is configured by setting pre-defined extend-attributes for files. Coding procedures will be invoked by file operations like read, write or set extend- attributes. The attributes indicate a file's coding scheme or an encryption mode. Different levels of encryption and encoding are provided to meet user's requirements. The details are shown in the table 1.

Table 1. Extend attributes and the file schemes

Attribute	Value	Algorithm
user.crsfs.rlevel	1	(128, 8)CRS
	2	(128, 16)CRS
	3	(128, 32)CRS
	4	(128, 64)CRS
user.crsfs.encrypt	1	AES-128-ECB
	2	AES-192-ECB
	3	AES-256-ECB
	4	AES-128-CBC
	5	AES-192-CBC
	6	AES-256-CBC

(k,m)CRS denotes Cauchy Reed-Solomon code with k data units and m checksum units. Packet size equals 512 bytes.

3.2 Kernel Functions

In this work, functions for both CUDA and OpenCL were implemented. For convenience, the CUDA implementation will be taken as an example for explanations. Details are given for the implementation of AES (Advanced Encryption Standard). As for the implementation of CRS (Cauchy Reed-Solomon) coding, it has been well described in the previous work [1] and is excluded from this paper.

Advanced Encryption Standard. To speed-up the execution, the parallel table lookup method is adopted [10]. The SubBytes, ShiftRows and MixColumns steps are combined by transforming them into a sequence of lookup tables. Each table contains 256 entries, where each entry has 32 bits. Tables are generated by the S-Box and the matrix on the MixColumns step. In a round, a 4-byte word is calculated by XOR operations with four table lookup results and a round key.

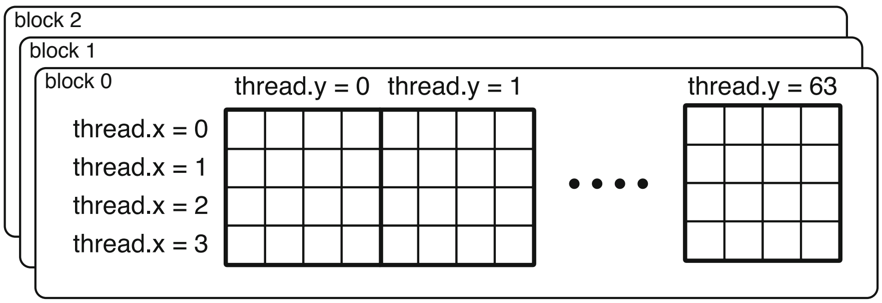


Fig. 1. AES kernel thread configuration

As for the threading configurations, it is illustrated in the figure 1. A block consists of 4x64 threads and processes 64 AES-blocks simultaneously. An AES block is handled by four threads that communicate to each other with shared-memories. In brief, a thread block is responsible for 1024 bytes, and the kernel will be launched with multiple blocks to process the data.

3.3 Runtime Framework

The runtime framework is designed to be the middleware between application and the computing platforms such as OpenCL and CUDA. It manages the dispatches of tasks and intends to maximize the throughput. Moreover, the framework is implemented by C++. It benefits from the characteristics of object-oriented programming. Several abstract classes and interfaces are defined.

The system overview of the runtime framework is illustrated in the figure 2. It consists of five major components, which are Task, Dispatcher, Worker, Stream and Kernel. Details are given below.

Task. Task is an abstract class that defines a job to be processed in the system. A task instance consists of the context of a job and has its own state maintained by the system. Functions are provided to manage tasks across the system. A sample code that encrypts a file is shown in the following listing.

```
// Create a set of tasks that encrypt a file.
TaskGroup *tasks = createFileEncryptionTasks(path, mode);
// Dispatch to the system asynchronously.
system->dispatchTasks(tasks);
// Block until the tasks are finished.
tasks.wait();
```

Dispatcher. Dispatcher assigns tasks to the workers. Functions can be overridden to apply other dispatch strategies. Tasks are pre-profiled by its characteristics and the experiment results. Based on the profiles, the dispatcher selects the most efficient way to process the tasks.

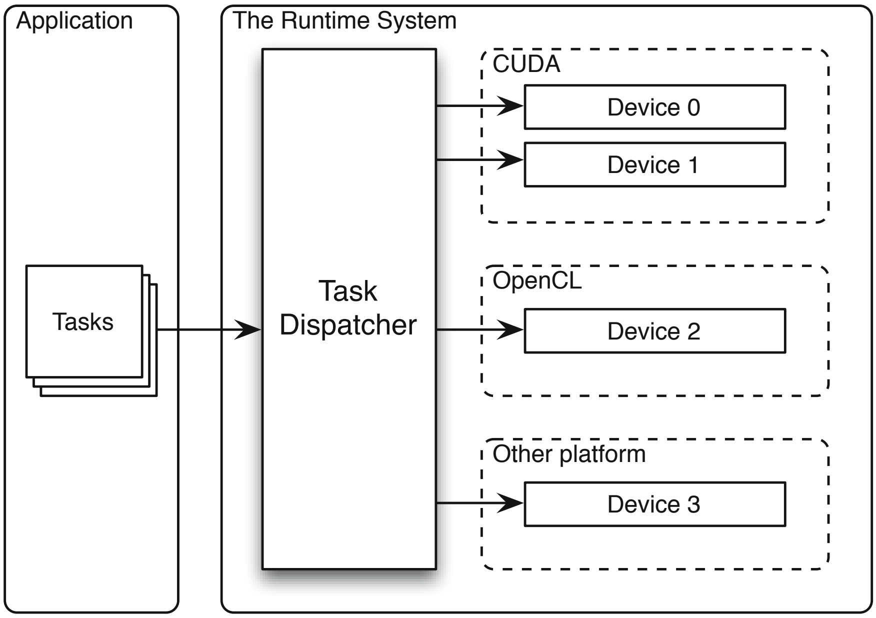


Fig. 2. Illustration of the runtime framework

Worker and Stream. Worker is a class that processes the tasks dispatched to. Each worker instance is associated with a hardware device, such as CPU or GPU, and it maintains its own task queue.

Also, a worker is created with multiple streams in order to achieve task-level parallelism within a device. Each stream is controlled by a CPU thread and consumes the pending tasks in the task queue. Therefore, for the traditional implementations equipped with multi-core CPU, the throughput can be improved by simply increasing the number of streams. As for GPU devices, multiple streams could pipeline the computation flows and improve the performance, as illustrated in figure.3.

Kernel. It is an abstract class associated with Worker and Stream. It defines the kernel functions that process the tasks. In order to run on different platforms, it is required to implement this class for each of them. Besides, the kernel implementations should also include other platform-specific operations, such as device initializations, resource allocations, kernel compilation and so on.

4 Evaluation

The evaluations include the runtime framework and the kernel functions. Performances on CPU, GPU, and a fusion system with multiple GPUs have been

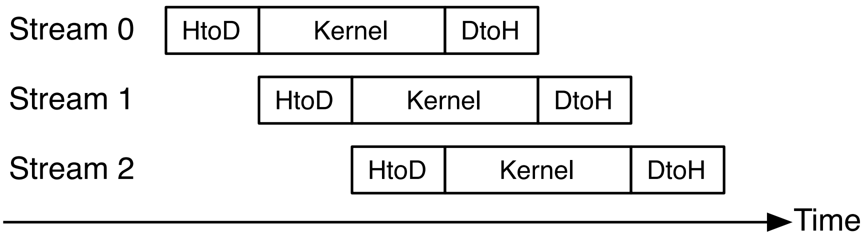


Fig. 3. Concurrency on GPU device
HtoD denotes the memory copies from host to device nd DtoH denotes memory copies from device to host.

tested and compared. Note that the evaluation of kernel functions will be focused on the AES operations. As for the CRS encoding/decoding routines, it has been well described in the previous work [1] and will be skipped.

The hardware configurations are listed in table 2 and the software settings are described in table 3. The computer used in the evaluation is equipped with two GPUs. The NVIDIA® Tesla™ K20c is used to execute the CUDA programs while the AMD Radeon™ HD 7970 is used to execute the OpenCL programs.

Table 2. Hardware specifications

CPU	Intel® Xeon® CPU E5620 @ 2.4GHz 4C/8T × 2
RAM	4GB DDR3-1066MHz × 6
GPU	Nvidia® Tesla™ K20c × 1
	AMD Radeon™ HD 7970 × 1
HDD	Hitachi 1.5TB 7200RPM × 2 (RAID 1)

Table 3. Software specifications

OS	Ubuntu 12.04.2
Kernel	Linux 3.2.0-38
File System	ext4
CPU Library	OpenSSL 1.0.1
CUDA	5.0
OpenCL	1.2 AMD-APP (1124.2)

4.1 Runtime Framework

Multiple situations are tested to see the impact on using different numbers of GPU streams. The evaluation is performed on CUDA platform with the Nvidia® Tesla™ K20c GPU. The test has two parts, which evaluate the performances of concurrent data transfer, and overlaps with kernel execution respectively.

Concurrent Data Transfer. A transfer task is defined for testing including two operations. It transfers the data from host to device, and then transfers back immediately. During the testing, multiple task instances are submitted into the system to measure the transmission throughput. The block size indicates the data size for each task instance.

As shown in the figure 4, it is evaluated by assigning different block size and different number of streams. For the result that use only one stream, the speed increases with the block size and reaches its maximum bandwidth at block size larger than 4MB. The peak performance is about 3GB/sec, limited by the PCIe hardware interface.

For the results with more than one stream, it benefits from the concurrent data transfer that the two transmissions may overlap. It is shown that the maximum speed is over 4GB/sec, having about 40% performance gain.

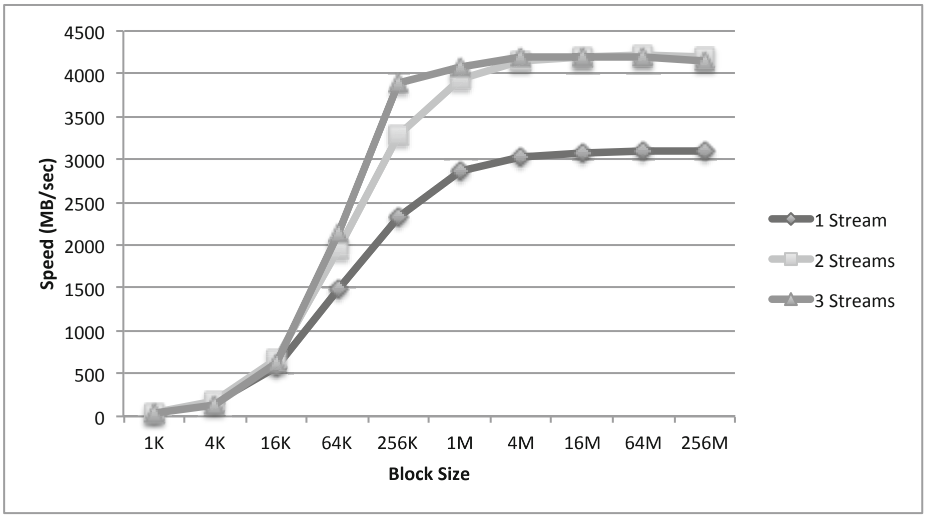


Fig. 4. Performances with different number of streams

Overlap with Kernel Execution. Then, the abilities of overlapping with kernel executions are evaluated. As listed in the table 4, two tasks including three operations: transferring data from host to device, launching a kernel function and transferring data from device to host, have been defined for testing. The kernel functions perform multiple floating-point operations on the data in parallel, both of the kernel execution times are approximately equals to three times the overhead of a single data transmission. The idea is that the task can be perfectly pipelined into five stages and have a five times performance gain. The first kernel possesses four thread-blocks, only occupy a small amount of computation resources. The device is capable to execute multiple kernel instances simultaneously. The second one possesses 1024 thread-blocks. During computation, the device is supposed to be fully occupied.

Table 4. Tasks for evaluating overlaps with kernel execution

Step	Operation	Time	Speed	Properties
1	Memcpy HtoD	664 us	6024 MB/sec	Data size = 4 MB
2	Kernel	1.87 ms	2139 MB/sec	Data size = 4 MB
				Grid size = 4
				Block size = 1024
3	Memcpy DtoH	629 us	6359 MB/sec	Data size = 4 MB

(a) Task A

Step	Operation	Time	Speed	Properties
1	Memcpy HtoD	669 us	5979 MB/sec	Data size = 4 MB
2	Kernel	1.88 ms	2128 MB/sec	Data size = 4 MB
				Grid size = 1024
				Block size = 1024
3	Memcpy DtoH	629 us	6359 MB/sec	Data size = 4 MB

(b) Task B

Next, the two tasks are tested by assigning different numbers of streams , as shown in the figure 5. For Task A, the speed increases quickly and reaches its maximum at using five streams. It is shown that the task is well pipelined by taking advantage of concurrent kernel executions. The speed is finally limited by the throughput of data transfer and is approximately 4GB/sec , which is consistent with the maximum speed shown in section 4.1. There is up to 3.25x performance gain compared to using only one stream.

As for Task B, the speed stops increasing at using two streams. Because the kernel have a great amount of thread-blocks that will fully occupy the GPU cores, only one kernel can be executed at the same time. The throughput is limited by the speed of kernel execution and is consistent with the kernel speed listed in table 5b. In this case, using multiple streams only benefits from overlaps with memory transmission and speeds up the throughput by 70%.

4.2 AES Operations

The data are wrapped into multiple AES-tasks and then being dispatched to the system. The block size is set to 1MB and the stream count is set to six for GPU devices.

Figure 6 shows the performance of AES operations on CPU with OpenSSL library. It is indicated that the execution time is directly proportional to the amount of CPU calculations, except for the very small size data. This result is used as a baseline for comparison.

Performance of AES Operations on GPU. Figure 7 shows the performance on CUDA. For data size under 16KB, the execution time is approximately a constant.

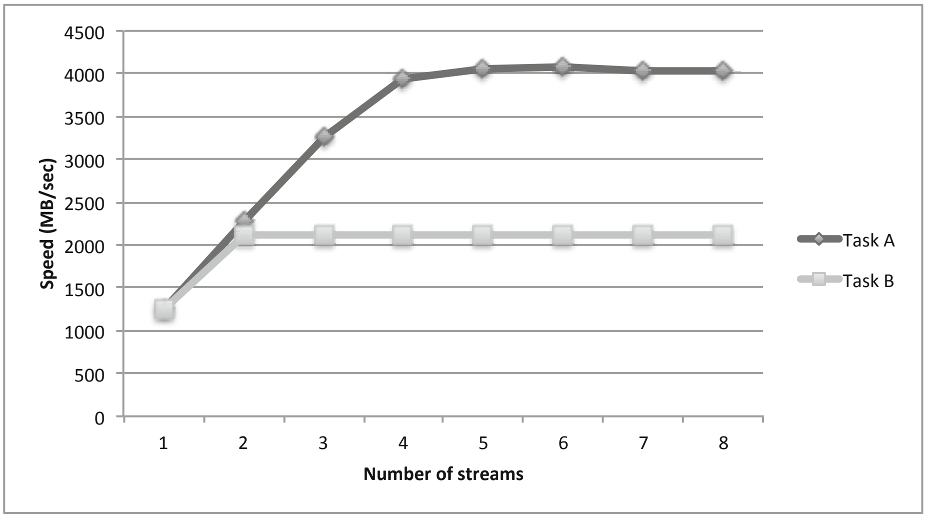


Fig. 5. Performances of evaluating overlaps with kernel execution

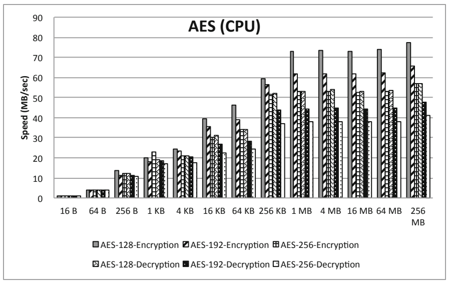


Fig. 6. Performances of AES operations on CPU

The overhead is introduced from the initializations of data transfer and launching kernels.

On the other hand, the speed increases significantly as the data size varies from 1MB to 4MB. It is caused by utilizing multiple streams. Note that the block size is set to 1MB. For data larger than 1MB, it is divided into 1MB-blocks and pipelined. The speed saturates at about 3.5GB/sec, which is closed to the maximum speed of data transfer tested in section 4.1. As for the performance on OpenCL, the trend is similar to CUDA, and the throughput saturates at about 1.6GB/sec.

Figure 8 shows the performance comparison between CPU and GPUs. For large data size, the results from using GPU are significantly better than the results from using CPU. The CUDA implementation has a 89.5x performance gain while the OpenCL implementation has a 41.2x performance gain.

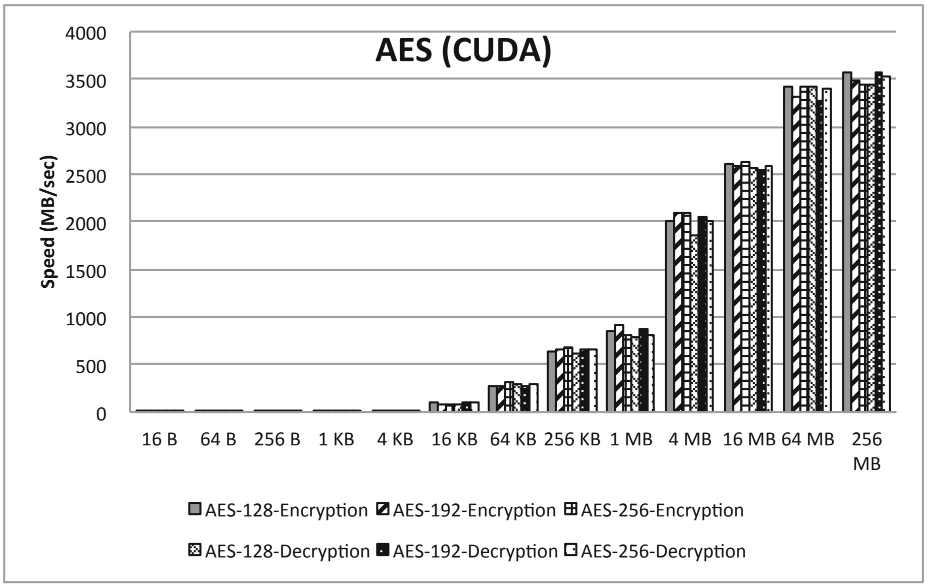
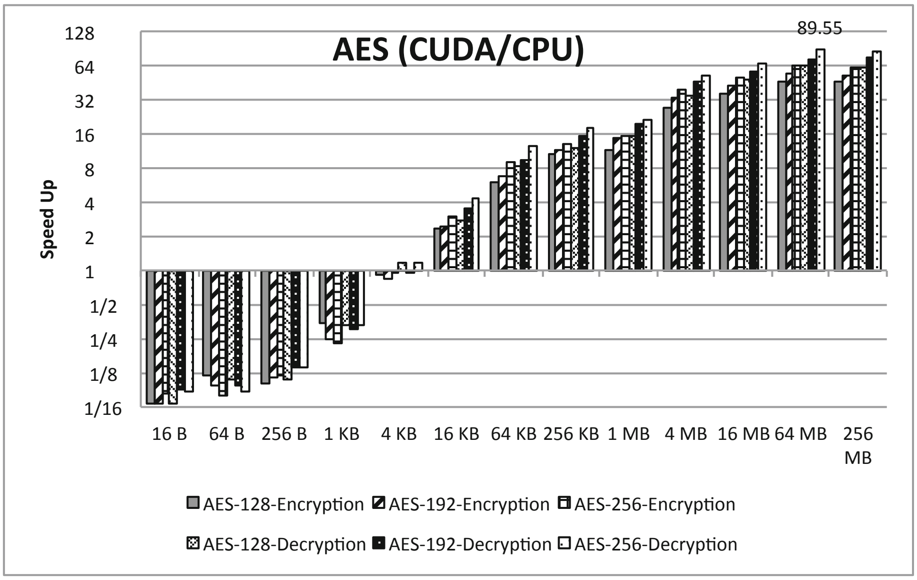


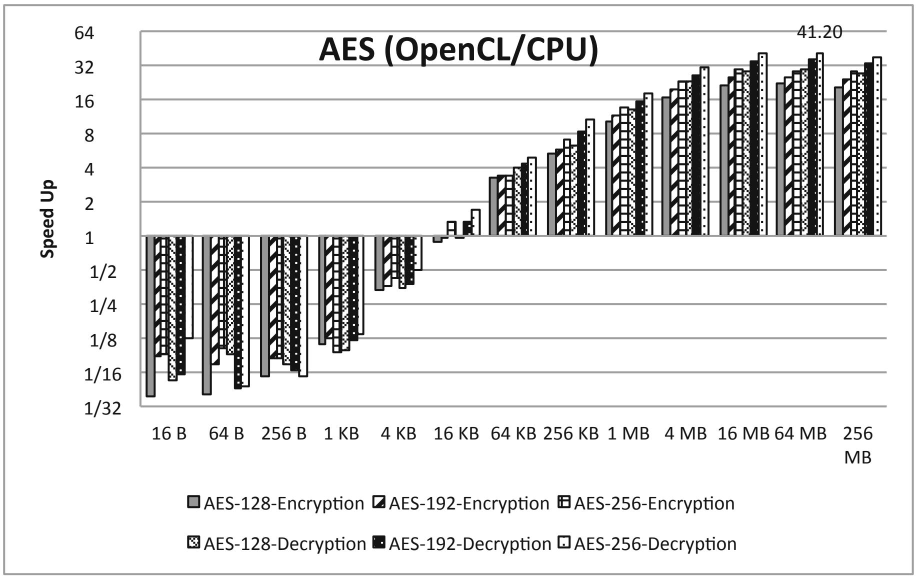
Fig. 7. Performances of AES operations on CUDA

Performance of AES Operations with Multiple GPUs. In order to have a better performance, the dispatcher assigns the tasks to different devices based on the task profiles from previous section. In this case , the strategy for processing AES operations is to dispatch the tasks with regard to the data size being processed. Details are described in table 5.

The performance comparisons of the fusion system are shown in figure 9. For large size data, the system throughput is increased up to 4.5GB/sec by using the two GPUs, and has a 104.57x performance gain compared to CPU implementation. Besides, there is no degradation on performance upon processing small size

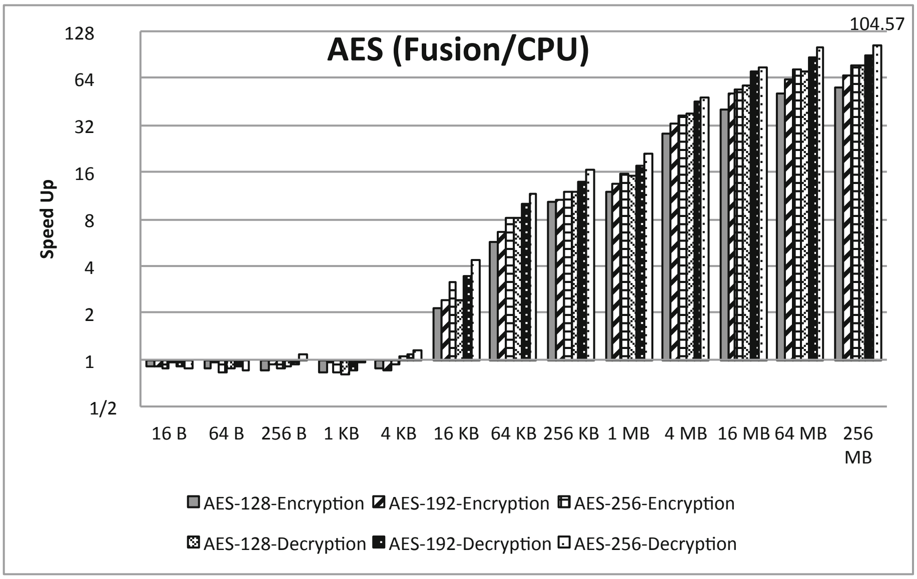


(a) Speed up with CUDA

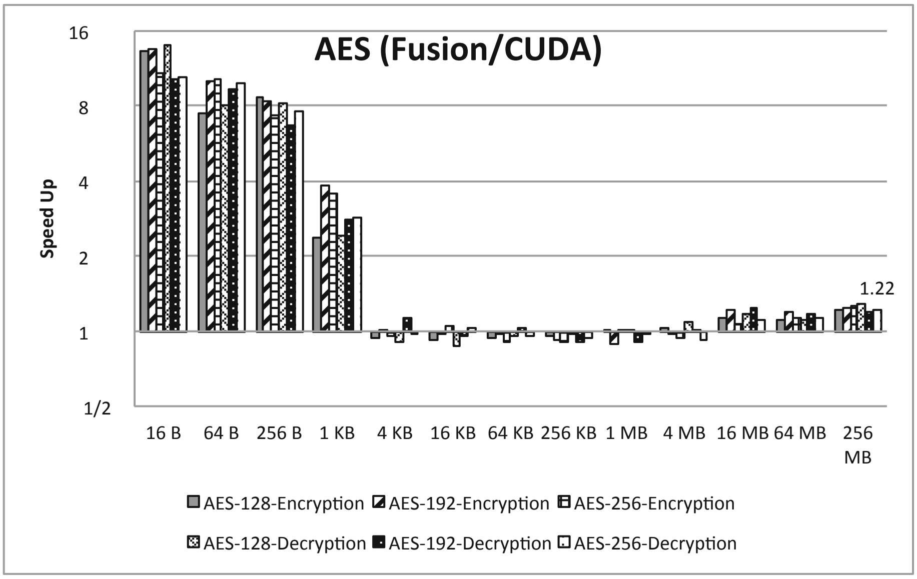


(b) Speed up with OpenCL

Fig. 8. Performances comparison of AES operations between GPUs and CPU



(a) Speed up with the fusion system



(b) Speed up compared to CUDA with the fusion system

Fig. 9. Performances comparison of AES operations with the fusion system

Table 5. AES tasks dispatch strategy

Data Size	Devices
$size<4KB$	CPU (Intel® Xeon® CPU E5620)
$4KB \leq size<16MB$	CUDA (Nvidia® Tesla™ K20c)
$size \geq 16MB$	CUDA (Nvidia® Tesla™ K20c) OpenCL (AMD Radeon™ HD 7970)

data. It is shown that the fusion system combines the benefits of each platform and processes the tasks in a more efficient way.

5 Conclusion

A runtime framework is proposed for heterogeneous computing, providing an infrastructure for developers to implement and optimize their system efficiently. The features include stream processing, a set of utilities for task managements, and the utilization of multiple devices. It is shown that the framework brings up to 40% performance gain on data transmissions and significant improvements in the overall throughput, varying from the application profiles.

In addition, a GPU accelerated file system prototype with enhancement of reliability and security is proposed. The original CRSFS is enhanced with the AES encryption features and is implemented with the new framework, which enables the file system to utilize more computation resources, including CUDA and OpenCL. The file system adapts a flexible design and runs as a stackable file system, allowing users to configure the coding schemes for each file and mount on any other existing file system.

Finally, it is shown that the AES operations can be improved by up to 89.5x using one Nvidia GPU(K20c) and 104.57x using two GPUs(Nvidia K20c and AMD HD7970).

6 Future Work

As there are more and more devices or platforms proposed for heterogeneous programming with different architectures or characteristics, such as the Intel® Xeon Phi™ co-processor or C++ AMP, the framework could be revised to have more integrations and optimizations for them.

In order to scale up the computation, deploying jobs to a distributed environment must be supported. It could combine the MPI interface to dispatch and manage tasks among the network.

On the other hand, the current dispatch strategies are profiled and optimized for a specific environment and application. It could be built with a task profiler which profiles task processing at the runtime, and then adjusts schedule policy instantaneously.

Acknowledgment The authors would like to thank the support from NSC under grants 102-2219-E-007-002 and 102-2220-E-007-003, and MOEA under grant 102-EC-17-A-02-S1-292.

References

1. Tseng, C., Lin, S., Hsu, Y.: A file system using gpu-accelerated file-wise reliability scheme (2012)
2. Blömer, J., Kalfane, M., Karpinski, M., Karp, R.M., Luby, M., Zuckerman, D.: An xor-based erasure-resilient coding scheme. ICSI, Tech. Rep. TR-95-048 (August 1995)
3. Gharaibeh, A., Al-Kiswany, S., Ripeanu, M.: Crystalgpu: Transparent and efficient utilization of gpu power. ArXiv preprint ArXiv:1005.1695 (2010)
4. Augonnet, C., Thibault, S., Namyst, R., Wacrenier, P.-A.: Starpu: A unified platform for task scheduling on heterogeneous multicore architectures. Concurrency and Computation: Practice and Experience 23(2), 187–198 (2011)
5. Aparapi: Api for data parallel java. Allows suitable code to be executed on gpu via opencl, `https://code.google.com/p/aparapi/`
6. Halcrow, M.A.: ecryptfs: An enterprise-class encrypted filesystem for linux. In: Proceedings of the 2005 Linux Symposium, vol. 1, pp. 201–218 (2005)
7. Sun, W., Ricci, R., Curry, M.L.: Gpustore: Harnessing gpu computing for storage systems in the os kernel. In: Proceedings of the 5th Annual International Systems and Storage Conference, p. 6. ACM (2012)
8. Le, D., Chang, J., Gou, X., Zhang, A., Lu, C.: Parallel aes algorithm for fast data encryption on gpu. In: 2010 2nd International Conference on Computer Engineering and Technology (ICCET), vol. 6, pp. V6-1–V6-6. IEEE (2010)
9. Iwai, K., Kurokawa, T., Nisikawa, N.: Aes encryption implementation on cuda gpu and its analysis. In: 2010 First International Conference on Networking and Computing (ICNC), pp. 209–214. IEEE (2010)
10. Murat Fiskiran, A., Lee, R.B.: Fast parallel table lookups to accelerate symmetric-key cryptography. In: International Conference on Information Technology: Coding and Computing, TCC 2005, vol. 1, pp. 526–531. IEEE (2005)

Efficient Detection of Cloned Attacks for Large-Scale RFID Systems

Xiulong Liu[1], Heng Qi[1], Keqiu Li[1,⋆], Jie Wu[2], Weilian Xue[3], Geyong Min[4], and Bin Xiao[5]

[1] School of Computer Science and Technology, Dalian University of Technology, China
keqiu@dlut.edu.cn
[2] Department of Computer and Information Sciences, Temple University
[3] School of Management, Liaoning Normal University
[4] Department of Computing, University of Bradford, UK
[5] Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Abstract. This paper studies a practically important problem of cloned tag detection in large-scale RFID systems where an attacker compromises genuine tags and produces their replicas, namely *cloned tags*, to threaten RFID applications. Although many efforts have been made to address this problem, the existing work can hardly satisfy the stringent real-time requirement, and thus cannot catch cloning attacks in a time-efficient way. Moreover, the existing work does not consider energy-efficiency, which is very critical when battery-powered active tags are used. To fill these gaps, this paper proposes a Location Polling-based Cloned tag Detection (LP-CTD) protocol by taking both time-efficiency and energy-efficiency into consideration. LP-CTD reports the existence of cloned tags when the reader finds an expected singleton slot appearing as collision one. To improve the efficiency of detecting cloned tags, only sampled tags in LP-CTD participate in the detection process. Theoretical analysis on the proposed protocol is conducted to minimize their execution time and energy consumption. Extensive simulation experiments are conducted to evaluate the performance of the proposed protocols. The results demonstrate that the proposed LP-CTD protocol considerably outperforms the latest related protocols by reducing more than 80% of the execution time, and more than 90% of the energy consumption.

Keywords: RFID Systems, Cloned Tag, Detection.

1 Introduction

Radio Frequency Identification (RFID) systems are widely used in various popular applications such as object tracking [1] [2] [3], supply chain management [4] [5], and access control [6] [7] owing to its attractive properties including remote and multiple access, as well as non-sight limitation. However, it is not easy to

⋆ Corresponding author.

prevent *tag cloning* attacks that threaten RFID-enabled applications [8]. The Cloned Tag Problem can be generally classified into two categories: (1) *cloned tag detection* problem, that aims to *detect* the existence of cloned tags as soon as possible [9]; (2) *cloned tag identification* problem, that aims to *exactly pin-point* which tags are cloned [10]. In practice, an RFID management system can first trigger a cloned tag detection protocol. Only when the detection result is affirmative will the identification protocol be invoked. A real RFID system may need to run the detection protocol periodically but seldom run the identification protocol. Designing a cost-effective cloned tag detection protocol becomes imperative when facing the threat of cloned tag problem. An straightforward way for cloned tag detection is to poll tag IDs one by one. A tag responds when its ID is queried. Clearly, any response collision can reveal cloned tag presence. However, polling ID is time-consuming because each tag ID is 96-bit long, especially when there is a large number of tags.

This paper studies the problem of cloned tag detection. Although RFID technology attracts much research interest, cloned tag detection is still in its infancy. The most promising *cloned tag detection* protocol is the Greedy collision-slot-REfrAming deTection (GREAT) protocol proposed in [9]. GREAT tackles this problem in anonymous RFID systems where the manager does not know the exact IDs of genuine tags. GREAT leverages unreconciled collisions to discover cloning attacks. An unreconciled collision (two or more tags always collide) is probably due to responses from multiple tags with the same ID, exact evidence of cloning attacks. However, in RFID applications [10] [11] [12] [13] that allow a backend to store the tag ID information, GREAT may has a room to be improved because it does not make use of the ID information. A long execution time renders the RFID system exposed to the threat of cloning attacks for a long time. Moreover, GREAT does not consider energy-efficiency, which is very important when battery-powered active tags are used to support advanced monitoring tasks in a large area. The frequent execution of this detection protocol will quickly drain the power of the active tags. Therefore, time-efficient and energy-efficient cloned tag detection protocols are still solicited.

In this paper, we propose a Location Polling-based Cloned tag Detection (LP-CTD) protocol by taking both time-efficiency and energy-efficiency into consideration. For the clarity of presentation, we first present two warm up protocols, which are the building blocks of the proposed LP-CTD protocol. Specifically, we first propose an Aloha-based Cloned tag Detection (A-CTD) protocol, in which the reader can predict the expected states of each slot and catches a cloning attack when an expected singleton slot turns out to be collision caused by the genuine tag and its replica(s). Based on A-CTD, we then exploit the *sampling idea* to propose a new Sampling-based Aloha-like Cloned Tag Detection (SA-CTD) protocol, in which only the sampled tags participate in the detection process, thereby achieving better time- and energy-efficiency. However, SA-CTD consists of many expected empty slots and collision slots, which contribute nothing to cloned tag detection and are wasted. Because the expected singleton slots are relatively rare in the frame, we propose LP-CTD to directly poll the index of expected singleton slots one by one. A tag will respond when its picked slot number is queried. After the index of an expected singleton slot is queried, if the reader receives a collision response, it asserts the existence of cloned tags. Besides the time-efficiency, the energy-efficiency is also considered in this paper. This paper theoretically investigates how to configure the system parameters of the proposed protocols in order to optimize their performance. Extensive simulation experiments are conducted to evaluate the performance of the proposed protocols. The results show that the proposed LP-CTD protocol significantly outperforms the latest related protocols, by reducing more than 80% of the required execution time, and more than 90% of the energy consumption.

The major contributions of this paper are summarized as follows:

1. This paper presents three protocols toward an efficient solution to the problem of cloned tag detection, where we take both of time-efficiency and energy-efficiency into consideration.
2. This paper presents theoretical analysis of the system parameters including the sampling probability and frame size on the performance of the proposed protocols to minimize their execution time as well as energy consumption.
3. Extensive simulation experiments are conducted to evaluate the performance of the proposed protocols. The experimental results demonstrate that the proposed protocols considerably outperform the latest related protocols.

The rest of this paper is organized as follows. The cloning attack model and the addressed problem are described in Section 2. The proposed A-CTD, SA-CTD, and LP-CTD protocols are presented in Sections 3-5, respectively. Simulation experiments are reported in Section 6 to evaluate the performance of the proposed protocols. This paper is concluded in Section 7.

2 Preliminaries

2.1 Cloning Attack Model

Consider an intact RFID system with a single reader and N genuine tags, where each genuine tag has a *unique* ID and is equipped with a common hash function $h(\cdot)$, whose result follows a uniform distribution. The genuine tag set is $T_{genuine} = \{tag_1, tag_2, ..., tag_N\}$. The reader has access to a database that stores the IDs of all N genuine tags [10] [11] [12] [13]. We adopt the same attacking model presented in the current literature [10] [9]. The attacker normally launches a cloning attack to RFID applications through three steps: (1) compromising genuine tags; (2) producing their replicas; (3) attaching cloned tags to unauthentic objects. Some genuine tags are cloned one or more times. Because the attackers require the cloned tags to behave the same as the genuine tags and can pass through any authentication as the genuine ones can, they clone not only the tag IDs but also the hash generator. Let us use M to denote the number of genuine tags that are compromised and cloned.

2.2 Problem Statement

This paper aims to detect the cloned tag event with a predefined accuracy, which is measured by two system parameters, a tolerance number m and a confidence level α. Inspired by [13] [14], the problem of cloned tag detection is defined as follows. *we desire to discover the cloned tag event with a probability of at least α when m (or more) genuine tags are compromised and cloned in the RFID systems.* For example, $\langle m = 5, \alpha = 99\%\rangle$ means that the objective is to discover the cloned tag event with a probability of 99% when 5 or more genuine tags are compromised and cloned. If a more accurate detection is expected, we may set a smaller m and a larger α.

2.3 Communication Overview

Communications between the reader and tags are made in a time-slotted way. The reader synchronizes the slot clocks of all tags by continuous control signals. If no tag replies in a slot, it is called an *empty slot*; if exactly one tag replies, it is called a *singleton slot*; and if two or more tags reply, it is called a *collision slot*. 1-bit tag response is enough to distinguish a empty slot from a busy one, but cannot distinguish singleton one from collision one. To distinguish a singleton slot from a collision one, the tag responses should be at least 10-bit. According to the specification of the Philips I-Code system [15], the wireless transmission rate from a tag to a reader is 53 Kb/s, that is, it takes a tag 18us to transmit 1 bit. And the rate from a reader to a tag is 26.5 Kb/s, that is, transmission of 1 bit to tags requires 37.7us. For simplicity, we assume the transmission rate from the reader to tags and that from tags to the reader are the same. We choose the relatively low rate 26.5 Kb/s as the common transmission rate. In [16], Li *et al.* presented a method of classifying the time slots: *tag slots*, *long-response slots* and *short-response slots*. The length of a tag slot is denoted as t_{tag}, which allows the transmission of a tag ID (96 bits), either from the reader to the tags or from a tag to the reader. The length of a long-response slot is denoted as t_{long}, which can afford to transmit a long response carrying 10 bits of information. The length of a short-response slot is denoted as t_{short}, which allows the transmission of a short response carrying only one bit of information. Since any two consecutive transmissions (from a tag to a reader or vice versa) are separated by a waiting time $\gamma_0 = 302us$, the time of a slot carrying ν-bit data, denoted as t_ν, is given by $t_\nu = \nu \times 37.7us + \gamma_0$. Hence, the length of the above three types of slots are described as follows. $t_{tag} = 96 \times 37.7us + \gamma_0 \approx 4ms$; $t_{long} = 10 \times 37.7us + \gamma_0 \approx 0.7ms$; $t_{short} = 1 \times 37.7us + \gamma_0 \approx 0.4ms$.

2.4 Performance Metrics

Given a specified detection accuracy that can be tuned to any accurate level, the time-efficiency is the most important performance metric—the shorter the execution time is, the sooner we will discover the cloning attacks and then conduct the corresponding countermeasures. The second important performance metric is energy-efficiency. To support advanced RFID applications that cover a large area, battery-powered active RFID tags are preferred, owing to their relatively longer communication range. In order to prolong the active tags' lifetime, we have to take the energy-efficiency into consideration.

3 An Aloha-Based Cloned Tag Detection Protocol

Based on the classic Slotted Aloha mechanism [17], we propose to utilize the expected singleton slots to detect the existence of cloned tags. We also investigate the configuration of the involved parameter settings and theoretically analyze the performance of the proposed Aloha-based Cloned tag Detection Protocol (A-CDP).

3.1 Protocol Design

The A-CDP protocol consists of two phases: *Slot Determination phase* and *Slotted Frame Execution phase*. The detailed procedures are described as follows.

Slot Determination Phase. The reader initiates this phase by broadcasting a binary $\langle R, f \rangle$, where R is a random number that is fresh in each execution and f denotes the length of the following slotted time frame. After receiving R and f, each tag then uses R, f, and its ID to determine a slot in the following frame by calculating a hash function $h(ID, R) \mod f$, whose result follows a uniform distribution within $[0, f-1]$. Because the reader knows all the involved parameters, the reader can predict which slots are expected to be singleton, empty or collision.

Slotted Frame Execution Phase. In this phase, a slotted time frame with f long-response slots is executed. A tag will respond in its determined slot. The reader listens and monitors the state of slots. If the reader finds that an *expected* singleton slot turns out to be collision, it asserts that the genuine tag corresponding to this slot is cloned, because this collision must be caused by the genuine tag and its replica(s).

3.2 Investigating the Configuration of Frame Length *f*

For a certain one of the M different compromised IDs, the probability that the reader can detect this ID as compromised (or cloned) is equal to the probability that this ID is mapped to an *expected singleton* slot. This probability, denoted as p_1, is given as follows:

$$p_1 = \binom{f}{1} \times \frac{1}{f} \times (1 - \frac{1}{f})^{N-1} \approx e^{-\frac{N}{f}}$$

When N is large, each of the M compromised IDs has the same probability p_1 to be detected. We use $P_A(N, M, f)$ to denote the probability that A-CTD can successfully detect the cloned tag event, when there are N genuine tags, M of them are compromised, and the frame length is f. $P_A(N, M, f)$ is equal to the probability that *at least one* of the M compromised IDs is detected, which is given as follows:

$$P_A(N, M, f) = 1 - (1 - p_1)^M = 1 - (1 - e^{-\frac{N}{f}})^M$$

Clearly, $P_A(N, M, f)$ is a monotonically increasing function with respect to M. Hence, $\forall M \geq m, P_A(N, M, f) \geq P_A(N, m, f)$. To meet the predefined detection accuracy, we only need to guarantee $P_A(N, m, f) \geq \alpha$. By solving this inequality, we have:

$$f \geq -N/(ln[1 - (1 - \alpha)^{\frac{1}{m}}]) \tag{1}$$

That is, any value of f larger than $-N/(ln[1 - (1 - \alpha)^{\frac{1}{m}}])$ can satisfy the predefined detection accuracy.

3.3 Performance of A-CTD

Time Cost. One *tag slot* t_{tag} is adequate for the reader to transmit the parameters R and f in the *Slot Determination phase*, and the time cost of the *Slotted Frame Execution phase* is $f \times t_{long}$. Hence, the total execution time of A-CTD, denoted as T_A, is given as follows:

$$T_A = t_{tag} + f \times t_{long} \approx f t_{long} \tag{2}$$

Energy Consumption Model. The energy consumed by the reader is of less concern because its battery can be easily recharged, or it may even use an external power source [12]. During a slot, an active tag has two types of states: the *awake* state and the *sleep* state [18]. Specifically, a tag needs to be in the awake state for communication, where the CPU operates at full energy and the radio remains open. Being awake, a tag may operate one of three actions: (1) transmitting data to the reader; (2) receiving data from the reader; (3) or just listening to the channel for receiving commands from the reader. Since the radio scanning consumes most of the energy, the above actions of an awake tag almost consume the same amount of energy. Hence, we simply use the time that a tag keeps active to measure its energy consumption. The energy consumption of the whole RFID system excluding the replica, denoted as E_{total}, can be given as follows:

$$E_{total} = \sum_{i=1}^{N} \tau_i \times \omega, \tag{3}$$

where τ_i is the time (in milliseconds) that tag t_i keeps awake for, and ω indicates the energy consumption of an awake tag per millisecond.

Energy Cost. In the *Slot Determination phase*, in order to receive the parameters $\langle R, f\rangle$, all the N genuine tags remain awake for a *tag slot*. The corresponding energy consumption is $Nt_{tag}\omega$. In the *Slotted Frame Execution phase*, for an arbitrary tag, it picks a slot from the frame following a uniform distribution. Let variable I denote the *index* of the slot that this tag selects. Clearly, $P(I = i) = \frac{1}{f} \mid i \in [1, f]$, then the expectation $E(I)$ is given as follows:

$$\begin{aligned} E(I) =& \sum_{i=1}^{i=f}[i \times P(I = i)] \\ =& 1 \times \frac{1}{f} + 2 \times \frac{1}{f} + 3 \times \frac{1}{f} +, \cdots, + f \times \frac{1}{f} = \quad \frac{1}{2}(f + 1) \end{aligned} \tag{4}$$

Hence, the energy consumption of this tag during the *Slotted Frame Execution phase* is expected to be $\frac{1}{2}(f + 1)t_{long}\omega$. For N genuine tags in total, the energy consumption during this phase is expected to be $\frac{1}{2}N(f + 1)t_{long}\omega$. We denote the energy cost of A-CTD as E_A which is given as follows:

$$E_A = Nt_{tag}\omega + \frac{1}{2}N(f + 1)t_{long}\omega \tag{5}$$

Clearly, both E_A and T_A are the monotonically increasing functions with respect to f. Hence, the proposed A-CTD protocol achieves the best energy-efficiency, as well as the best time-efficiency, when f is minimized to $-N/ ln[1 - (1 - \alpha)^{\frac{1}{m}}]$.

4 A Sampling-Based Aloha-Like Cloned Tag Detection Protocol

The A-CTD protocol proposed in last section needs *all* tags participate in the detection process, which consumes a vast amount of energy. Moreover, when there is a large number of tags in the system, the frame size should be set very long in order to guarantee there are enough expected singleton slots to expose the cloned tags. Intuitively, if we can discover the existence of cloned tags by performing detection on just a small number of sample tags, a more efficient cloned tag detection protocol could be achieved. Inspired by [14], we exploit the sampling idea to propose a Sampling-based Aloha-like Cloned Tag Detection (SA-CTD) protocol.

4.1 Protocol Description

Here, we exploit the sampling process proposed in [14], which is described as follows. The reader broadcasts a request, which consists of two parameters: a random number R_1 and an integer $x = \lceil p \times X\rceil$, where p is the sampling probability and X is a large constant pre-configured in RFID tags. After receiving the request, each tag performs a hash function $h(ID, R_1) \mod X$, which follows a uniform distribution within $[0, X)$. If the hashing result is less than the received parameter x, the tag will participate in the following process; otherwise, it will enter the sleep state and will not participate in the following detection process [14]. Note that, if a compromised genuine tag is sampled, its cloned peers (i.e., replica tags) will also be sampled because they have the same ID information. Since all the above sampling decisions are made pseudo-randomly depending on the used parameters, the reader can predict all the decisions and know exactly which genuine tags are sampled. Then, we perform A-CTD on the sampled tags to detect the cloning tag event.

4.2 Investigating the Configuration of the Sampling Probability *p* and the Frame Length *f*

For a certain one of the M compromised IDs, the probability that the reader can detect this ID as compromised (or cloned) is equal to the probability that this ID is sampled *and* mapped to an expected singleton slot. We denote this probability as p_2, and it is given as follows:

$$p_2 = p \times [\binom{f}{1} \times \frac{1}{f} \times (1 - \frac{p}{f})^{N-1}] \quad \approx pe^{-\frac{Np}{f}},$$

where p is the sampling probability and f is the length of the slotted frame. We use $P_{SA}(N, M, p, f)$ to denote the probability that the proposed SA-CTD protocol can successfully discover the cloned tag event when there are N genuine tags, and exactly M of them are compromised and cloned; the sampling probability is p; the frame length is f. $P_{SA}(N, M, p, f)$ is equal to the probability that *at least one* of the M compromised IDs is detected, and can be given as follows:

$$P_{SA}(N, M, p, f) = 1 - (1 - p_2)^M \quad = 1 - (1 - pe^{-\frac{Np}{f}})^M$$

Similarly with the analyzes of A-CTD, we only need to guarantee $P_{SA}(N, m, p, f) \geq \alpha$ for meeting the required accuracy. By solving this inequality, we have $f \geq -Np/(\ln[\frac{1-(1-\alpha)^{\frac{1}{m}}}{p}])$. Obviously, a longer time frame consumes more time; given a fixed sampling probability p, the frame length f should be minimized to $-Np/(ln[\frac{1-(1-\alpha)^{\frac{1}{m}}}{p}])$. Note that, if the sampling probability p is too small, less than $1 - (1 - \alpha)^{\frac{1}{m}}$, the frame size f will become a *negative* value. Such a small sampling probability cannot be used. Thus, the smallest sampling probability p_{min} is larger than $1 - (1 - \alpha)^{\frac{1}{m}}$.

4.3 Analyzing the Performance of SA-CTD

Time Cost. Compared with prior A-CTD protocol, the proposed SA-CTD needs an extra tag slot to broadcast the used parameters. Hence, the *total* execution time of the proposed SA-CTD protocol, denoted as T_{SA}, is given as follows:

$$T_{SA} = 2t_{tag} + ft_{long} \approx ft_{long} \qquad (6)$$

Energy Cost. In the *Sampling phase*, all the N genuine tags have to remain awake for *one tag slot* in order to receive the parameters $\langle R_1, x\rangle$. The corresponding energy consumption during the sampling phase is $Nt_{tag}\omega$. Np (expectation) sampled genuine tags and their replicas (if there are any) will remain awake and participate in detection process which is the same as that of A-CTD. It is easy to deduce that the *total* energy cost of SA-CTD, denoted as E_{SA}, is given as follows:

$$\begin{aligned} E_{SA} &= Nt_{tag}\omega + Np \times t_{tag}\omega + Np \times [\frac{1}{2} \times (f+1)t_{long}\omega] \\ &= N(1+p)t_{tag}\omega + \frac{1}{2}Np(f+1)t_{long}\omega \end{aligned} \tag{7}$$

The SA-CTD protocol outperforms the prior A-CTD protocol in terms of both time-efficiency and energy-efficiency. There is a sampling probability p_t that minimizes the time cost of SA-CTD (i.e., operating at the time-saving mode), and a sampling probability p_e that minimizes the energy cost of SA-CTD (i.e., operating at the energy-saving mode). Note that, p_t and p_e can be easily obtained through many offline methods (e.g., differentiation, exhaustive search, etc.).

5 A Location Polling-Based Cloned Tag Detection Protocol (LP-CTD)

The SA-CTD protocol proposed in the last section still has a room to be improved. Specifically, SA-CTD leverages the expected singleton slots to detect the cloned tags. However, a large number of expected empty slots and expected collision slots contribute nothing to the detection of cloned tags, which leads to the low time-efficiency. An immediate challenging issue is how to avoid the execution of expected empty slots and expected collision slots. In this section, we first propose a Location Polling-based Cloned tag Detection (LP-CTD) protocol. Then, we theoretically investigate the parameter settings to optimize the performance of LP-CTD.

5.1 Protocol Description

After the sampling process that is the same as that of SA-CTD, we propose to directly poll the *index* of the expected singleton slots one by one. And a tag responds only when its selected slot number is queried. By this method, the expected empty/collision slots are skipped. Specifically, the reader first broadcasts the query $\langle R, f\rangle$ to all tags. Each tag selects a slot by calculating $h(ID, R)$ mod f. Because the reader knows the employed hashing function as well as the used parameters, it is able to predict which slots are supposed to be singleton. Then, the reader queries the tags by broadcasting the index ($\lceil log_2 f\rceil$ bits long) of the expected singleton slots one by one. Upon receiving a polling request, a tag check if the index of its selected slot matches the queried one. If they exactly match, the tag will respond. On the reader side, if it senses a collision slot after broadcasting the index of an expected singleton slot, the tag corresponding to this index must be compromised.

5.2 Investigating the Configuration of the Sampling Probability *p* and the Frame Length *f*

The fundamental difference between LP-CTD and SA-CTD is that LP-CTD avoids the execution of expected empty/collision slots by directly polling the index of expected singleton slots, which actually does not affect the detection accuracy. That is, the same as SA-CTD, given fixed p and f, the frame size f should be larger than $-Np/(\ln[\frac{1-(1-\alpha)^{\frac{1}{m}}}{p}])$. But the optimization of p and f of LP-CTD is different from that of SA-CTD.

5.3 Analyzing the Performance of LP-CTD

Time Cost. Similar to the SA-CTD protocol, two tag slots are adequate to transmit the initialization parameters. In the *polling phase*, each slot in the virtual frame has the following probability to be singleton.

$$P_s = \binom{N}{1} \times \frac{p}{f} \times (1-\frac{p}{f})^{N-1} \approx \frac{Np}{f}e^{-\frac{Np}{f}}, \tag{8}$$

The number of expected singleton slots, denoted as η_s, follows Bernoulli (f, P_s). Hence, $E(\eta_s) = f \times P_s = Npe^{-\frac{Np}{f}}$. Each expected singleton slot corresponds to two parts of time costs: (i) time for querying the index with length of $\iota = \lceil log_2 f \rceil$ bits; (2) time for tag responding. Therefore, the execution time of LP-CTD, denoted as T_{LP}, is given as:

$$T_{LP} = 2t_{tag} + E(\eta_s) \times (t_\iota + t_{long}) \quad \approx Npe^{-\frac{Np}{f}}(t_\iota + t_{long}) \tag{9}$$

Energy Cost. In the sampling phase, all N tags keep awake for a tag slot to receive the used parameters. The corresponding energy consumption is $Nt_{tag}\omega$. In the slot determining phase, Np awake tags (expectation) receive the initialization parameters. The corresponding energy consumption is $Npt_{tag}\omega$. In what follows, we analyze the energy consumption in the polling phase. For a certain tag, we use the conditioning probability $P\{S|L\}$ to denote the probability that there are S expected singleton slots proceeding when the tag is mapped to the L^{th} slot in the virtual frame. $P\{S|L\}$ is given as follows.

$$P\{S = j|L = i\} = \binom{i}{j}(P_s)^j(1-P_s)^{i-j}, j \in [0, i] \tag{10}$$

The corresponding conditioning expectation $E(S|L = i)$ is given as follows.

$$\begin{aligned} E(S|L = i) &= \sum_{j=0}^{i}[j \times P\{S = j|L = i\}] \\ &= \sum_{j=0}^{i}[j \times \binom{i}{j}(P_s)^j(1-P_s)^{i-j}] \\ &= \sum_{j=1}^{i}[i \times \binom{i-1}{j-1}(P_s)^j(1-P_s)^{i-j}] \\ &= iP_s\sum_{j=1}^{i}[\binom{i-1}{j-1}(P_s)^{j-1}(1-P_s)^{i-j}] = iP_s \end{aligned} \tag{11}$$

The expectation $E(S) = \sum_{i=0}^{f-1} E(S|L=i) \times P\{L=i\} = \sum_{i=0}^{f-1}(iP_s \times \frac{1}{f}) = \frac{f-1}{2}P_s$. That is, there are $\frac{f-1}{2}P_s$ expected singleton slots proceeding the slot that this tag selects. The selected slot has the probability $(1-\frac{p}{f})^{N-1} \approx e^{\frac{Np}{f}}$ to be an expected singleton one, if so, the tag will receive the queried index and respond to the reader. On the other hand, the selected slot has the probability $1-e^{\frac{Np}{f}}$ to be expected collision, if so, this tag has to receive the queried index of the next expected singleton slot before knowing it picks an expected collision slot and entering sleep state. The expected energy consumption of this tag, denoted as ζ, is given as follows.

$$\begin{aligned} \zeta &= E(S) \times (t_\iota + t_{long})\omega + (t_\iota + t_{long})\omega \times e^{\frac{Np}{f}} + t_\iota\omega \times (1-e^{\frac{Np}{f}}) \\ &\approx \frac{1}{2}Npe^{-\frac{Np}{f}}(t_\iota + t_{long})\omega \end{aligned} \tag{12}$$

For Np sample tags in total, the energy cost of LP-CTD, denoted as E_{LP}, is given as follows:

$$E_{LP} = Np \times \zeta = \frac{1}{2}N^2p^2e^{-\frac{Np}{f}}(t_\iota + t_{long})\omega \tag{13}$$

6 Performance Evaluation

6.1 Simulation Setting

In this section, we evaluate the performance of the proposed protocols via simulators. Following most of the RFID related studies [19] [20], we also consider a single reader in the simulations, and assume that this reader has adequate power to interrogate with a large number of RFID tags via a error-free wireless channel. Each simulation is conducted 1000 times, and we get the average results.

6.2 Investigating the Impact of m and α

In this subsection, we investigate the impact of the system parameters m and α on the performance of the proposed protocols. The number N of genuine tags is fixed to $20,000$; the tolerance number m varies from 20 to 40; the detection accuracy α varies from 90% to 99%. The simulation results illustrated in Fig. 1 indicate that a small tolerance number m (or a large confidence level α) will increase the energy cost, as well as the time cost of the proposed protocols. Clearly, the LP-CTD protocol considerably outperforms the other two protocols (i.e., A-CTD and SA-CTD) in terms of both time-efficiency and energy-efficiency.

6.3 Comparing with the Latest Related Protocols

In this subsection, we compare the performance of LP-CTD (our final protocol) with two latest related protocols about cloned tag problem, i.e., the GREAT protocol and the ES-BID protocol, under different values of N, m, and α. The

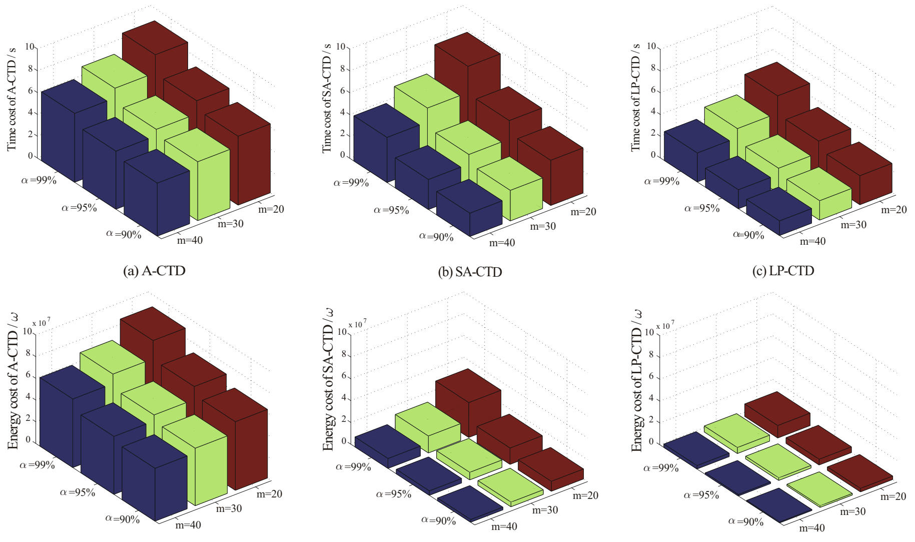
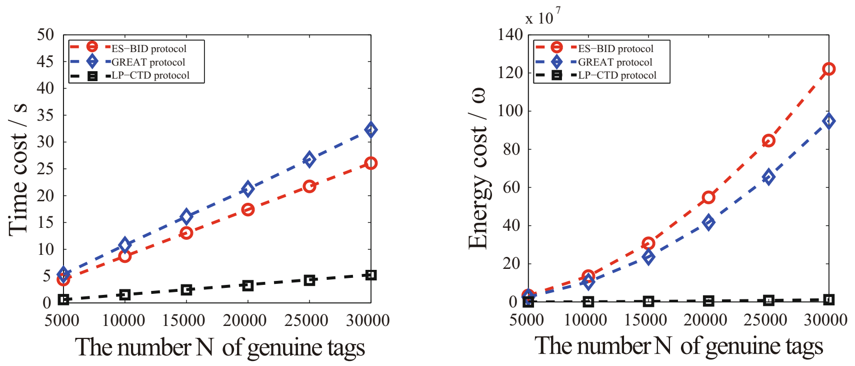


Fig. 1. Investigating the impact of m and α on the performance of the proposed protocols, where the number N of genuine tags is fixed to $20,000$

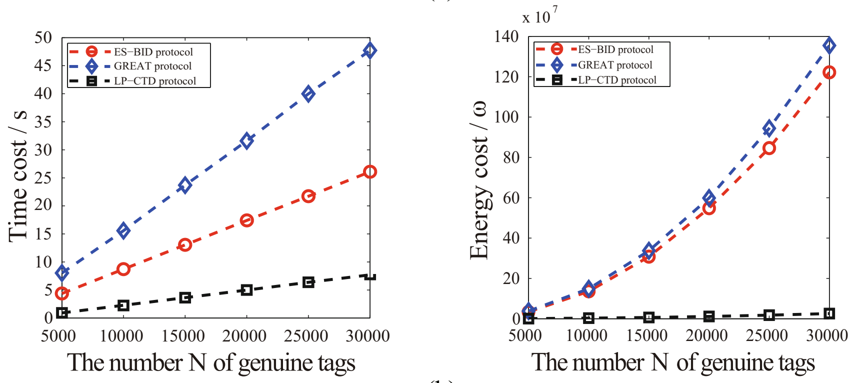
simulation results shown in Fig. 2 demonstrate that the proposed LP-CTD protocol considerably outperforms GREAT and ES-BID in terms of both time-efficiency and energy-efficiency. For example, when $N = 30,000$, $m = 20$, and $\alpha = 95\%$, the time cost of GREAT and ES-BID is 32.3 s and 26.1 s, respectively. The time cost of the proposed LP-CTD is just 5.2 s, representing 83.9% and 80.1% reduction in terms of time-efficiency when compared with GREAT and ES-BID. Under the same parameters, the energy cost of GREAT and ES-BID is 94.84×10^7 ω and 122.14×10^7 ω, respectively. The energy cost of the proposed LP-CTD is just 1.18×10^7 ω, representing a 98.8% and 99.0% reduction in terms of energy-efficiency when compared with GREAT and ES-BID.

6.4 Evaluating the Real Detection Accuracy

This subsection evaluates the *real detection accuracy* of the proposed LP-CTD protocol, where we simulated m (or more) compromised tags among all N genuine tags. The results shown in Table 1 and Table 2 demonstrate that LP-CTD could well satisfy the pre-defined detection accuracy α. For instance, given a high security requirement: $m = 1$; $\alpha = 99\%$, the *real detection accuracy* is higher than the required 99% when $1 \sim 4$ tags are compromised; given a relatively relaxed security requirement: $m = 10$; $\alpha = 95\%$, the *real detection accuracy* is higher than the required 95% when $10 \sim 13$ tags are compromised.



(a)



(b)

Fig. 2. Comparing LP-CTD with the ES-BID protocol and GREAT protocol under different N, m and α. (a) $m = 20$, $\alpha = 95\%$; (b) $m = 20$, $\alpha = 99\%$.

Table 1. A high security requirement: $m = 1$; $\alpha = 99\%$. M varies from 1 to 4.

M compromised tags	1	2	3	4
Total Test Runs	10,000	10,000	10,000	10,000
Successful Detection	9,907	9,999	10,000	10,000
Failed Detection	93	1	0	0
Real Detection Accuracy	99.07%	99.99%	100.00%	100.00%

Table 2. A relatively relaxed security requirement: $m = 10$; $\alpha = 95\%$; M varies from 10 to 13

M compromised tags	10	11	12	13
Total Test Runs	10,000	10,000	10,000	10,000
Successful Detection	9,505	9,643	9,738	9,780
Failed Detection	495	357	262	220
Real Detection Accuracy	95.05%	96.43%	97.38%	97.80%

7 Conclusion

This study investigates the practically important problem of cloned tag detection, where we take both time-efficiency and energy-efficiency into consideration. This paper presents a series of protocols toward an efficient solution to the studied problem. Theoretical analysis is conducted to minimize the execution time as well as energy consumption. Extensive simulation experiments are conducted to evaluate the performance of the proposed protocols. The results demonstrate that the proposed LP-CTD reduces more than 80% of the required execution time and more than 90% of the required energy consumption, when compared with the latest related protocols. In this paper, we consider the error-free communication channels, which is a general assumption in most of RFID-related literature. In our future work, we will conduct experiments on the real RFID platforms to investigate the impact of channel errors and propose the countermeasures.

Acknowledgement. This work was supported by NSFC (Grant Nos. 60973117, 61173160, 61173162, 60903154), New Century Excellent Talents in University (NCET) of Ministry of Education of China, The Research Fund for the Doctoral Program of Higher Education (No. 20130041110019) and the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61225010).

References

1. Nemmaluri, A., Corner, M.D., Shenoy, P.: Sherlock: Automatically locating objects for humans. In: Proc. of ACM MobiSys (2008)
2. Kodialam, M., Nandagopal, T., Lau, W.C.: Anonymous Tracking using RFID tags. In: Proc. of IEEE INFOCOM (2007)
3. Matic, A., Papliatseyeu, A., Osmani, V., Mayora-Ibarra, O.: Tuning to Your Position: FM radio based Indoor Localization with Spontaneous Recalibration. In: Proc. of IEEE PerCom (2010)
4. Lee, C.H., Chung, C.W.: Efficient storage scheme and query processing for supply chain management using RFID. In: Proc. of ACM SIGMOD (2008)
5. Luo, W., Qiao, Y., Chen, S.: An Efficient Protocol for RFID Multigroup Threshold-based Classification. In: Proc. of IEEE INFOCOM (2013)
6. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication. Wiley (2010)
7. Nath, B., Reynolds, F., Want, R.: RFID technology and applications. In: Proc. of IEEE PerCom (2006)
8. Juels, A.: RFID Security and Privacy: A Research Survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
9. Bu, K., Liu, X., Luo, J., Xiao, B., Wei, G.: Unreconciled Collisions Uncover Cloning Attacks in Anonymous RFID Systems. IEEE Transactions on Information Forensics and Security, TIFS (2012)
10. Bu, K., Liu, X., Xiao, B.: Fast Cloned-Tag Identification Protocols for Large-Scale RFID Systems. In: Proc. of IEEE IWQoS (2012)

11. Shahzad, M., Liu, A.X.: Every Bit Counts - Fast and Scalable RFID Estimation. In: Proc. of ACM MobiCom (2012)
12. Luo, W., Chen, S., Li, T., Qiao, Y.: Probabilistic Missing-tag Detection and Energy-Time Tradeoff in Large-scale RFID Systems. In: Proc. of ACM MobiHoc (2012)
13. Tan, C.C., Sheng, B., Li, Q.: How to Monitor for Missing RFID tags. In: Proc. of IEEE ICDCS (2008)
14. Luo, W., Chen, S., Li, T., Chen, S.: Efficient Missing Tag Detection in RFID Systems. In: Proc. of IEEE INFOCOM (2011)
15. Semiconductors, P.: I-CODE Smart Label RFID Tags (January 2004), `http://www.nxp.com/acrobat_download/other/identification/SL092030.pdf`
16. Li, T., Chen, S., Ling, Y.: Identifying the Missing Tags in a Large RFID System. In: Proc. of ACM MobiHoc (2010)
17. Roberts, L.: ALOHA packet system with and without slots and capture. ACM SIGCOMM Computer Communication Review 5(2), 28–42 (1975)
18. Chlamtac, I., Petrioli, C., Redi, J.: Energy-Conserving Access Protocols for Identification Networks. IEEE/ACM Transactions on Networking 7, 51–59 (1999)
19. Qiao, Y., Chen, S., Li, T., Chen, S.: Energy-efficient Polling Protocols in RFID Systems. In: Proc. of ACM MobiHoc (2011)
20. Yue, H., Zhang, C., Pan, M., Fang, Y., Chen, S.: A Time-efficient Information Collection Protocols for Large-scale RFID Systems. In: Proc. of IEEE INFOCOM (2012)

Probability Based Algorithms for Guaranteeing the Stability of Rechargeable Wireless Sensor Networks

Yiyi Gao, Ce Yu*, Jian Xiao, Jizhou Sun, Guiyuan Jiang, and Hui Wang

School of Computer Science and Technology, Tianjin University, Tianjin, China
JaelGao@163.com, yuce@tju.edu.cn

Abstract. The lifetime of a wireless sensor network can be prolonged by introducing mobile sink to balance the energy consumption in data gathering. However, it doesn't solve the bottleneck of energy consumption and network will still fail inevitably. The option of recharging a relay opens up new possibilities for prolonging the network lifetime even maintaining the network living forever by recharging a sensor using a mobile sink when necessary. In this paper, we investigate the strategies for the mobile sink to recharge the wireless network in order to maintain the stability of network achieving the requirement of network system. In our approach, we develop an efficient algorithm to divide the sensor network into a number of domains (independent communication areas which are consisted by a cluster of neighbored sensors) in the first stage. Then we iteratively calculate the maximum lifetime of each domain. When travelling and arriving at a domain, mobile sink determines whether or not recharge it according to the probability model. The lifetime of a domain is estimated based on the residual energy of each sensor within the domain using our energy routing algorithm. We also propose a probability model for the mobile sink to determine whether to recharge a domain or not. The effectiveness of our approach has been verified by extensive simulation results.

Keywords: Wireless sensor network, network lifetime, energy routing, probability model.

1 Introduction

The wireless sensor networks (WSNs) consist of a stationary based station (BS) and hundreds to thousands of sensors, which collect the information from the near environment by short-range communication and transmit them to BS by some routing strategies. WSN is widely used in the fields of monitoring, target tracking and etc. Based on the sensors' working function, the traditional WSNs can be classified into rechargeable WSNs and non-rechargeable WSNs.

*Corresponding author.

In traditional non-rechargeable WSN, all the sensors hold limited energy and can not be recharged once deployed in the experiment, while the based station can be regarded as unlimited energy infrastructure with power generation equipment (solar generator etc). The sensors communicate with BS in the function of multi-hops routing, where part of sensors serve for forwarding other sensors' information. As continually sensing, receiving and transmitting information, the sensors will use up all the initial energy that can not work any more, where the sensor is called dead. In practice, many applications using WSN as large systems make some important decisions based on all the information from difficult sensors deployed in the area of interest. In other word, only one sensor's death will effect the whole system. Therefore, we say that the WSN's death is at the same time of the first sensor's death in the network and define the period of WSN working from deployed to death as the network's lifetime. Due to the different distance to BS and different amount of forwarding, the energy consumption of each sensor has a huge gap. The unbalanced energy consumption among sensors shortens the network lifetime and effect other network performance seriously.

To cope with the drawback, recent research for non-rechargeable WSN is to deploy a mobile sink travelling from the BS and collecting sensors' information in the areas of interest[1,2,3]. Thus, the energy consumption of each sensor is balanced through the motion of sink. Many experiments and simulations demonstrate the effectiveness of the strategy in terms of different network performance. In traditional non-rechargeable WSNs, all the sensors are battery-powered, that means no matter adopting any efficient strategy, the power of any one sensor will be dissipated over and the WSN will die.

In traditional rechargeable sensor networks, the sensor has a special ability of energy harvesting that means catching energy from the experiment, such as solar, wind and so on[4]. Through the network lifetime is not the bottleneck of WSN any more, the network has highly dependence in the environment and the sensors generally are hard to be deployed due to the nature of the sensors.

Considering those drawbacks in rechargeable and non-rechargeable sensor network, this paper strives to ensure the stability of network and pursue a appropriate tradeoff between the energy consumption of mobile sink and network lifetime. The function of recharging is only by mobile sink other rather the energy harvesting in traditional rechargeable WSNs, and the mobile sink have unlimited energy as it can get back to BS for recharging.

The main contributions of this paper are as follows. Firstly, we propose an algorithm for efficiently partitioning the network into disjoint domains and find a resultful recharging tour. Secondly, jointly given the load-balancing and network lifetime, we design a routing strategy based on residual energy and an algorithm for estimating the lifetime of domain. Finally, a novel model probability model for network is proposed, which can ensure the stability of the network as much as possible by changing the parameters of this probability model.

The remainder of the paper is organized as follows. Section II summarizes the related work in both rechargeable WSN and non-rechargeable WSN. Section III introduces the system model, energy model and defines the problem precisely.

Section IV proposes three algorithms: Graph Partition algorithm, Maximal Domain Lifetime Estimation (MDLE) and Maintenance of Network. Section V evaluates and analyzes the performance of the proposed algorithm through simulational experiments, and Section VI concludes the paper and introduce the further works.

2 Related Work

Extensive studies on optimizing network resources in WSN with mobile sink have been conducted in the past few years. The main performance metrics are network lifetime, energy consumption, the number of rendezvous nodes and etc. Many researches work on the tradeoff among different network performance under some constraints, such as [5]. In conventional network with only one mobile sink, the battery-powered sensors are used to achieve the experimental information and transmit them to the sink for analysis and management. The general communication model between sensors and sink is multi-hop routing, which is tree topology. Extensive experiments and simulations have been done in this network model, such as[6,7]. The tree topology model has a serious drawback, that is the energy consumption of the sensors near to the sink have increased rapidly due to the more data transmission. Previous experiments and simulations prove that when the 1-hop neighbors of sink dies, the energy of sink remains 93% of initial state[12]. Firstly, using the method of Steiner trees[9,10], we can plan the trajectory of mobile sink better and solve RN replacement problem, which can be widely used in network recovery. Secondly, in routing strategy based on energy, [6,11] combine the shortest path tree and minimum spanning tree for prolonging the network lifetime. Finally, some researches adopt the method of maximum flow for prolonging the network lifetime[13]. The three methods above aim to prolong the network lifetime and save energy consumption in traditional wireless network with a mobile sink. However, no matter how much they reduce the energy consumption, the network will die, as the sensor can only hold limited energy and can't be recharged once deployed in the environment. Given the RN(rendezvous node) has an ability to be charged by mobile sink[4], the sensors can also be recharged by sink. In actually, the sensor is most based on battery, which means that the consideration is realistically feasible. The purpose of this paper is to maintain the network more stable (the network lifetime is above the standard of requirement) with less energy consumption.

3 Preliminaries

3.1 System Model

A wireless sensor network consists of a large number of low-powered sensor nodes for sensing and caching the experimental information and a mobile sink which has unlimited energy. A WSN can be modeled as an undirected, connected graph $M = (N \bigcup \{s\}, L)$, where N is a set of n stationary, identical sensors randomly deployed in an interest area, s is the mobile sink and L is the set of links between sensor u and sensor v (MS is regarded as special sensor) if and only if they are within the transmission range(R) of each other. In this paper, sensors are considered identical as they have the same sensed range, transmission range and energy-carrying capacity. Defined by $N_1(u)$, the set of 1-hop neighbors of sensor u means the set of possible sensors which can directly communicate with sensor node u. $N_1(u) = \{v|(u,v) \in L\}$. Defined by $N_h(u)$,the set of neighboring sensors of the sensor node u in h-hops, $N_h(u) = \{$v$|u$ communicates with v at least in h hops$\}$. Given any sensor node u,the calculation of $N_h(u)$ is as follow. A Breath-First-Search tree rooted as u is constructed layer by layer. We define the $R_h(u)$ as the neighbors of u within h-hops. $R_h(u) = \{v|(v,w) \in L, w \in N_{h-1}(u)$ and v not in $R_{h-1}(u)\}$. It is assumed that all the sensors have the same initial energy ME and the mobile sink with unlimited energy supply can sojourn at any location for data gathering. In WSN with mobile sink, the tour-length of trajectory of the mobile sink affects the total energy consumption. For shortening the tour-length, we divide the graph into domains, which consist of few sensors and have autonomy by rendezvous node (RN: a sensor with special task). The RN can aggregate the sensed data from the sensor in its domain, then analysis them and make appropriate response, so it is called that the RN can manage the domain or the sensors in this domain. The RN has enough large buffer so that can cache all the sensed data in its domain before the mobile sink comes. By that means, we reduce the number of sensors to be travelled, thus the tour-length will be shorten. In each partitioned domain, sensors are working in the model of multi-hops routing, whose topology is a tree, as balanced as possible. Every node in tree has depth, which it is called level in the paper. RN's Level 0 and the level of other sensors are defined as the least hops to RN dominating them.

The paper will attempt to design the model for the motion and working of sink, design a strategy of graph partition and data gathering in inn-domain. In order to formally define the problem, we introduce the following notations and variables.

- The energy consumption of a sensor includes the computation, sensing, data transmission and data reception. As all the identical sensors alive have the approximate energy consumption for sensing environment, we only consider the energy of data transmission and reception in this paper. Let k_{tran},k_{recv} represent the energy consumed by a sensor node for transmitting and receiving one bit respectively and k_{ampt} denotes the consumption for amplifying a bit's signal.
- For a given data transmission or reception, we consider that the size of sensed data by any sensor is identical to a fixed length l
- For a sensor with residual energy, β is a self-adapting variable of sensor in specified domain.

Definition 1. The network lifetime is defined as the first sensor failure time due to its energy depletion. The lifetime of each domain is the time span from the partition calculated to the instant when the domain is considered nonfunctional.

The moment when a domain is nonfunctional is the instant when the first sensor node dies (can't transmit data).

3.2 Energy Model

It is well known that the energy consumption is a significant metrics in WSN. Premature death of a node can interrupt the communications and results in the death of the network. The energy model involving the energy consumption parameters is necessary. To transmit or receive a data session (a fixed length l) between sender to receiver (where the distance between them is d), the energy consumed can be expressed as:

$$E_{tran} = l \times k_{tran} \times d + l \times k_{ampt} \times d^2$$
$$E_{recv} = l \times k_{recv}$$

We denote the residual energy of a sensor u by RE_u, which is a significant metrics for the routing in the next part.

3.3 Residual Energy Routing

In general multi-hops routing, the energy is rapidly dissipated in the sensors near to the sink. The main cause is that the sensors nearby the sink does not only transmit its own sensed data but also is used for relaying and forwarding the message from higher-level sensors.

Definition 2. The weight cost of a link (u, v), for a transmission from node u to node v, where node u 's level is always lower than node v, is defined as below:

$$C_{uv} = min\{RE_u - E_{tran}, RE_v - E_{recv}\}$$

Definition 3. For the sensors u in Level 1,when its residual energy is less than β times to C_{uv} (v is the RN), it turns into low-powered state. Once a sensor turns into low-power, it will not work as relay node but only transmit its own sensed data to RN.

As the C_{uv} is the significate variable with the ability of prediction, in each turn of calculation, we choose the sensor v in lower Level with the maximum C_{uv} as the next hop. Then update all the C_{uv} for the next turn of calculation.

4 The Probability Based Algorithm

In the scheme of data collection with mobile sink (MS),while the sensor cache their data from environment nearby and send them to rendezvous nodes (RN) via short-range transmission, the mobile sink travels in the interest areas only to collect the data from RNs (a RN caches all the data of local sensors managed by itself). On the consideration that MS has an ability of recharging,our problem is defined as follows:

Problem. *Given a set of sensors N , mobile sink s and a relay hop bound d, find:*

- *A set of RNs, which cover all the sensor within d-hops. A set $Domain_i$ of sensors covered by RN_i within d-hops, $Domain_i \bigcap Domain_j = \emptyset$, $\bigcup RN_i = RNs$.*
- *$Lifetime_i$ of $Domain_i$ is the time span from $Domain_i$ built to any sensor failed.*
- *For given parameters p and α, a tour U for MS to collect data or recharge some domains, such that the network lifetime is above the standard of network requirement and the total length of MS's motion (Euclidean distance) is minimum.*

The ultimate goal is select appropriate parameters p and α to meet the stability of the network requirements. Corresponding to the above three parts of problem, we will introduce three algorithms in detail, which are Network Partition, Domain Lifetime Estimation and Network maintenance respectively.

4.1 Network Partition

Network partition plays a important role in rendezvous node based data gathering scheme. Unbalanced partition leads to unbalanced energy consumption among domains, thus some domain will run out of energy much earlier that others and result in the death of the entire network. In order to prolong the life time of the entire network, the number and distribution of domains must be jointly considered.

On the other hand, in each domain, one sensor node is selected as rendezvous node to cache data from its nearby neighbors vis a tree based routing path. Thus, the depth of the routing tree should be restricted to a certain level. We define $B_h(u)$ as the set of uncovered sensors that can communicate with u at least in h hops.

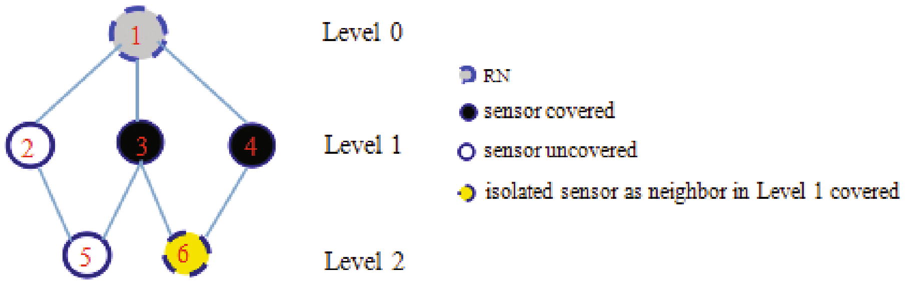


Fig. 1. Choose neighbors to cover for graph partition

$B_h(u) = \{v \,|(v, w) \in L, w \in B_{h-1}(u), v \text{ is uncovered } \}$

We can get $B_h(u)$ using an breadth-first search technique. As an example given in `Fig 1`, $B_1(RN) = 2$ and $B_2(RN) = \{5\}$. As sensor 2 and sensor 3 have been covered, they can not be members of the domain managed by RN. Sensor 6 also can not be a member of this domain, because its neighbors in Level 1 have all been covered by other sensors and RN can not get its data packet from any sensors in Level 1 in current situation.

As the routing for data collection in each domain is in the form of tree topology, the sensors nearby the RN of each domain consume energy much faster than other sensors. Therefore, these sensors affect the lifetime of its domain. Let $Domain_s$ be a domain rooted at RN s, the domain fails to function when all sensors in $N_1(s)$ run out of energy. For the whole network, the more sensors in Level 2 it has, the more energy consumption it dissipates in specified time. Thus, we use $t_h(u)$ to balance the stability for the domain being partitioned, where $t_h(u) = |B_1(u)| \setminus \sum_{i=2}^{h} |B_i(u)|$. For the example in Fig. 1, the $t_2(u) = 1 \setminus 1 = 1$. The domain managed by the sensor u with maximum $t_h(u)$ has larger probability for maintaining stabilization. While the topology information of network is input, *DomainSet* is outputted as the set of domains partitioned, the algorithm for network partition in detail is as follows:

Algorithm 1. Network Partition

```
Require: $G(N \bigcup \{s\}, L), h$
Ensure: $DomainSet$
 1: Find a sensor node $u$ in $G$ ,which has the largest Euclidean distance from
    sink $s$
 2: while the sensors in $G$ aren't all covered do
 3:     Find a uncovered sensor node $u$ in $R_h(s)$,which has the largest $t_h(u)$
 4:     Cover $\sum_1^h B_i(u)$ by node $u$
 5:     $DomainSet \leftarrow domain(\sum_1^h B_i(u) \cup u, L')$
 6:     if $B_{h+1}(u) = \emptyset$ then
 7:         $s \leftarrow$Find the uncovered node with longest Euclidean distance from $s$ as
            new $s$
 8:     else
 9:         $s \leftarrow$Find the uncovered node in $B_{h+1}(u)$ with longest Euclidean distance
            from $s$
10:     end if
11: end while
12: return $DomainSet$
```

Due to the strategy we choose in inn-domain, the depth effect domain's lifetime directly. In this paper, we all default the 2-hops's parameter to do the Network partition and Domain lifetime estimation.

4.2 Domain Lifetime Estimation

Based on the `Algorithm 1`, we have got all the partitioned domains. In this part, a strategy will be proposed for estimating the domain's lifetime. Due to the parameter of 2-hops partition, the domain we calculate has three different hierarchies: Level 0, Level 1 and Level 2. The sensor in Level 0 is the rendezvous node(RN) in this domain, the sensors in Level 2 are only working as conventional sensors for experimental information gathering and in additional, the sensors in Level 1 can be used as relay for forwarding the higher level's data session when they are not in low-power state.

In the parse of network working, the data generation(the size of data sensed by any sensor in specified time) can be considered as a const. A turn is defined by the times for RN to gather all the sensors' data in specified timespan. In the paper, the number of turns from the partition built to being invalid is used as the domain's lifetime. A domain is considered as invalid, when (1)it has died node (2)when one turn is over, the RN hasn't gathered all the sensors' data.

A domain has its own parameter β to maintain the maximum lifetime. Whether a sensor is in the state of low-power or not is due to that parameter. The β is achieved by several trying. Once it is achieved, it will keep the value same to maintain the network stability in the next parse.

In `algorithm 2`, we consider a sensor died when it can't transmit its own data to its lower-level neighbor and the update working is based on the changed RE_u and RE_v. The detail algorithm design is as Algorithm 2 below.

Algorithm 2. Domain Lifetime Estimation

```
Require: Domain $S$,$\beta$
Ensure: $lifetime$
 1: while RN gets all the sensors' information do
 2:     for all the sensors in Level 1 do
 3:         sensor $u$ update its energy using the energy consumption of transmitting
            data to RN
 4:         mark sensor $u$ has transmitted its data
 5:     end for
 6:     Find sensor $u$ in Level 2 and non-low-power sensor $v$ in Level 1 with the
        maximum $C_{uv}$
 7:     mark sensor $u$ has transmitted its data
 8:     Update the $C_{uv}$
 9:     if Any sensor $w$ in domain is Died then
10:         break;
11:     end if
12:     lifetime++
13: end while
14: return lifetime
```

In conventional WSN with a mobile sink, the lifetime of network is the time span from sensors deployed to the first sensor's power-off. Further, the network's lifetime is equal to the minimum lifetime of domains. In next part, we will discuss the network's lifetime in the case of WSNs recharged by a MS.

4.3 Network Maintenance

In conventional WSN with a mobile sink, most researches focus on the reducing the length of sink's trajectory for energy saving. However, this function can't prolong the network. As current sensors can store energy, we do a significant suppose that the mobile sink can recharge every sensor when sojourning at the location of them for gathering their data in this paper. Like conventional methods, we plan the trajectory of the sensor's motion based on TSP(Travelling Salesman Problem) for shortest tour-length among RNs.

Algorithm 3. Network Maintenance

```
Require: $domain\_u$,$tour\_length$,$speed$,$p$,$\alpha$
Ensure: $netlife$
 1: while no died sensor in the whole network do
 2:    for Each domain $u$ in $S$ do
 3:       $dominator \leftarrow u$'s $RN$
 4:       $lifecycle = (tour\_length - u.lasttime)/speed$
 5:       the domain run $lifecycle$ times
 6:       if there is any sensor died in the domain then
 7:          break;
 8:       else
 9:          if there is any sensor low-powered in the domain then
10:             recharge the $domain\_u$ with probability $p$
11:          end if
12:       else
13:          recharge the $domain_u$ with probability $\alpha$
14:       end if
15:    end for
16:    $netlife$++;
17: end while
18: return $netlife$
```

Given the limited lifetime of domains, they need to be recharged at appropriate time, thus the tour-length of sink also has necessity to take the length of TSP in inn-domain into account. Due to the speed for data gathering, the MS could not recharge every sensor in one turn of its motion. Thus, a mechanism for mobile sink's working will be designed as follow.

(a)When a mobile sink sojourn at the location of RN, if the domain has low-power sensor, the mobile sink recharge the domain based on the probability of p

(b)Once the mobile sink arrives at RN, if the domain has no died or low-power sensor, the mobile sink recharge the domain based on the probability of α

(c)At the arrival of mobile sink, if the domain already has died sensor, then the whole network is died.

The probability model of mobile sink's working is due to the parameters p and α, which are up to the users. In our design, as the sink has an ability of recharging sensors, we don't only consider the network lifetime, but pursue for the less energy consumption under the network's stability. We use the metric of mobile sink's tour-length from network working into its being invalid to measure the performance of energy consumption. By this way, we can achieve appropriate value of the parameters by numerical experiments.

For computing the life cycle consumed of each domain, the RN must record the mobile's total tour-length and compare to the value of sink's last arrival.Based on the difference between the tour-lengths of those two times and the speed of mobile sink, we can calculate the life cycle consumed in the span between sink's two arrivals. In Algorithm 2, we have already calculate all domain's lifetime, and then

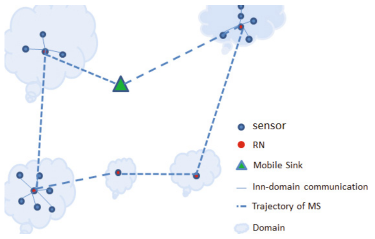
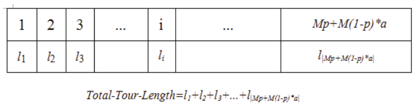


Fig. 2. A tour for the mobile sink to visit each domain



1	2	3	...	i	...	*Mp+M(1-p)*a*
l_1	l_2	l_3		l_i		$l_{\|Mp+M(1-p)*a\|}$

Total-Tour-Length$=l_1+l_2+l_3+\ldots+l_{|Mp+M(1-p)*a|}$

Fig. 3. Sort of inn-domain tour-length of domains

estimate the network lifetime in conventional methods. For given the parameters p and α, the average of the domains to be recharged is $Np + N\alpha(1 - p)$, where N is the number of domains. We add the tour-length of TSP among RNs with the $Np + N\alpha(1 - p)$ longest tour-length of TSP in inn-domains as Total-Tour-Length, where N is the number of domains. The radio of Total-Tour-Length divided by network lifetime is considered as the mobile sink's speed, which defined as Specified-Speed. We bring in a parameter γ and the real speed of sink is γ times to Specified-Speed for the estimation life cycle of each domain.

l is the tour-length of TSP among RNs and l_i is the tour-length of TSP with the ith longest trajectory in inn-domain.The detailed algorithm design is as follow.

`Algorithm 3` will return life cycles of the network as the maximum network lifetime by $netlife$.

In the algorithm above, the ME is a sensor's maximum energy capacity while the tour-length is sink's total moving distance from the networking till now.

In total, our algorithm is divided into three steps. Firstly, we calculate the graph partition with the potential lifetime, then do estimation of lifetime for each partitioned domain. Finally, we maintain the network working and obtain the appropriate parameters by numerical experiments in practical applications.

5 Experiments and Analysis

In this section, we evaluate the efficiency of the proposed algorithms and present the simulation results. The performance metrics are mainly the residual energy of every sensors of a domain in and the stability of network under different p and α, where p represents the probability of recharging a domain in the state of low powered and α stands for the one of recharging a living domain which is not lower-powered.

5.1 Simulation Environment

In the simulation, we consider a generic sensor network with N sensors randomly distributed over an $L * L$ square area. The mobile sink s is located in the area randomly. The transmission range of a sensor is R. Based on the above network model, each packet is locally aggregated to a RN within 2 hops. As the standard of network stability is decided by users, we classify the stability of network into four different states as `Table 2`. The least lifetime of network corresponding to different stability is listed in `Table 2`. Considering the randomness of the network topology, the result of each case is averaged over the results for a random simulation instance running 20 times.

5.2 Results and Analysis

For a domain in a running WSN (the current network lifetime is 400, p=0.7, α=0.4) with 100 sensors randomly deployed in 200*200 areas, `Fig 4` illustrates the residual energy of each sensors in a domain, while `Table 1` represents the communication relationship among them. We can explicitly observe from `Fig 4` that balanced residual energy have been achieved among all sensors of the domain. Considered the sensors' communication relation given by `Table 1`, we find that the 6th sensor has no neighbors in Level 2. In other words, the 6th sensor in this domain is used only for sensing the environment and forwarding its own data to RN but working as relay. The load for sensors in Level 1 working as relays are balanced, so the routing strategy based on residual energy is efficient for avoiding the premature death of network.

`Fig 5` shows the degree of network stability under different p and α, where p ranges from 0.1 to 1.0 while α ranges from 0.01 to 0.99 increased by 0.01. As the high randomness of the model itself, for a given p, we measured the stability of network by the number of feasible resolutions under some constraints from experiments with different value of α. From `Fig 5`, we can observe that the network becomes very stable when p is above 0.5.

`Fig 6` shows the network stability affected by α. Similarly with the `Fig 5`'s experiment, for a given α, the stability of network is represented by the number of feasible resolutions from experiments with different value of α. `Fig 6` shows the α has a high randomness for the effect on the stability of network. In actual, the value of α depends on the topology of initial graph itself. Thus, it is necessary for users to adjust this parameter by many simulational experiments.

Table 1. Communication relationship among sensors

	Level2-1	Level2-2	Level2-3	Level2-4	Level2-5	Level2-6	Level2-7
Level1-1	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$
Level1-2	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$
Level1-3	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Level1-4	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Level1-5	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Level1-6	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
Level1-7	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Levelj-i: the ith sensor in Level j.
$\surd$ at location (i,j) means the Level1-i can communicate with Level2-j
$\times$ at location (i,j) means the Level1-i can communicate with Level2-j

Combining `Fig 5` and `Fig 6` together, the stability of network is jointly effected by p and α. Both the growth of p and α could enhance the stability of network in some range. When p is lower, the stability of network must be maintained by a with an enough high value. Due to the high randomness of α, the stability of network can not increase consistently as the growth of p when p is lower than 0.5, which is clearly shown by `Fig 6`.

Table 2. Parameters for experiments

ME(sensor's energy capacity)	0.5J
k_{ampt}	100pJ/bit
k_{tran}	50nJ/bit
k_{recv}	50nJ/bit
l	1000bit
acceptable stability	4000
high stability	3500
sub-high stability	3000
acceptable stability	500

Similarly, `Fig 7` shows the high randomness of α effecting the total length of MS's trajectory. Through the length of MS's trajectory per lifetime (LPT) is increasing positively with α's growing ,the total length of MS's trajectory depends on both the network lifetime and LPT. Due to the significant randomness of α, the total length also shows the randomness. So, there is much necessity to adjust α for a specific environment, which has an unique topology structure.

Seeking for the more stable WSN and less energy consumption of MS, we must take the parameters p and α into consideration jointly. For different requirement for stability, we need to choose proper p first and then adjust the value of α for higher stability and less energy consumption.

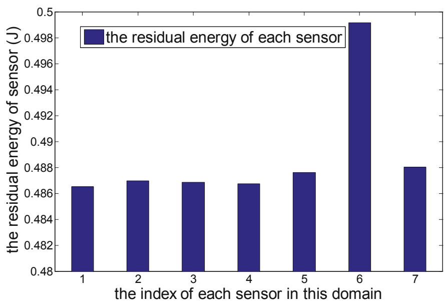


Fig. 4. Residual energy of sensors

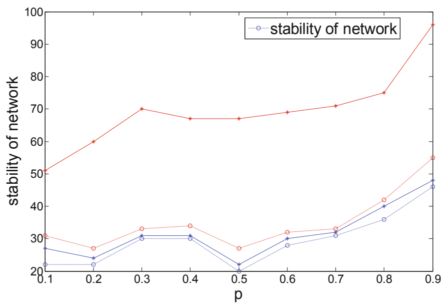


Fig. 5. Network stability with different p

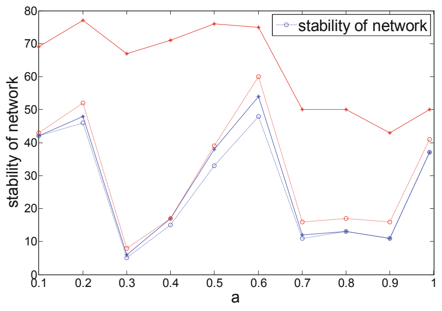


Fig. 6. Network stability with different α

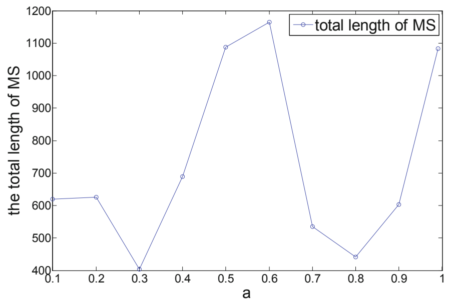


Fig. 7. Total length of MS's trajectory with different α

6 Conclusion and Future Work

In the paper, we investigate the problem of prolonging the lifetime of rechargeable wireless sensor networks, using a mobile sink to recharge low power sensors when necessary. To find out the most efficient recharge order and recharge tour, we partition the entire sensor networks into many disjoint domains. We propose approach to estimate the maximum lifetime of each domain so that domains with short lifetime must be recharged in order to avoid power-off. We propose a probability model for the mobile sink to decide whether to recharge the sensors of a domain or not. Energy can be saved by selecting proper parameters of the probability model. In addition, the model has the ability of self-learning to adapt the interest areas. The effectiveness of our approach has been verified by extensive simulation results. In the future, we plan to do the following work: We want to adopt other methods to replace this routing tree structure taking the too much energy consumption of nodes in Level 1. For example, we can dynamically choose the sensor as RN for collecting the domain's sensed data.

Acknowledgment. The work is sponsored by the National Natural Science Foundation of China (61303021).

References

1. Liang, W., Schweitzer, P., Xu, Z.: Approximation Algorithms for Capacitated Minimu Forest Problems in Wireless Sensor Networks with a Mobile Sink. IEEE Transactions on Computers 62(10) (2012)
2. Buragohain, C., Agrawal, D., Suri, S.: Power Aware Routing For Sensor Database. Proceedings of INFOCOM (2005)
3. Cristescu, R., Beferull-Lozano, B., Vetterli, M.: On Network Correlated Data Gathering. In: Proceedgins of INFOCOM (2004)
4. Vieira, M.A.M., Coelho, C.N.: Survey on Wireless Sensor Network Devices. In: Emerging Technologies and Factory Automation, vol. 1, pp. 537–544 (2003)
5. Bari, A., Teng, D., Jaekel, A.: Optimal Relay Node Placement in Hierachical Sensor Networks with Mobile Data Collector. In: Global Telecommunications Conference, pp. 1–5 (2011)
6. Feng, S., Liang, W., Luo, J., Xu, X.: Network lifetime maximation for time-sensitive data gathering in wireless sensor network. Computer Networks 11(1), 47–60 (2012)
7. Liu, W., Fan, J., Zhang, S., Wang, X.: Relay Hop Constrained Rendezvous Algorithm for Mobile Data Gathering in Wireless Sensor Networks. In: Hsu, C.-H., Li, X., Shi, X., Zheng, R. (eds.) NPC 2013. LNCS, vol. 8147, pp. 332–343. Springer, Heidelberg (2013)
8. Tan, H.O., Korpeoglu, I.: Power Efficient Data Gathering and Aggregation in Wireless Sensor Network. SIGMOD Record 32(4), 66–71 (2003)
9. Faith, S., Mohamed, Y.: Optimized Connectivity Restoration in a Partitioned Wireless Sensor Network. Information Processing Letters 69, 53–57 (2011)
10. Izzet, F.S., Kemal, A., Rith, S., Mohamed, Y.: Connectivity Restoration in Disjoint Wireless Sensor Networks using Limited Number of Mobile Relays. IEEE Transactions on Mobile Computing 11(1), 47–60 (2012)
11. Khuller, S., Raghavachari, B., Neal, Y.: Balancing Minimum Spanning and Shortest Path Trees. IEEE Transactions on Mobile Computing 11(1), 47–60 (2012)
12. Wadaa, A., Olariu, S., Wilson, L., Jones, K., Xu, Q.: On Training a Sensor Networks. In: Proc. of Parallel and Distributed Processing Symp. (2003)
13. Ataul, B., Da, T., Arunita, J.: Maximum Flow Based Model and Method of the Maximum Lifetime Problem of Sensor Network. In: Intelligent Control and Automation, pp. 21–23 (2006)
14. Sharaf, M.A., Beacer, J., Labrinidis, A., Chrysanthis, P.A.: Balancing Energy Effcientcy and Quality of Aggregate Data in Sensor Networks. Journal of VLDB (2004)

PTAS for Minimum *k*-Path Connected Vertex Cover in Growth-Bounded Graphs

Yan Chu, Jianxi Fan[*], Wenjun Liu, and Cheng-Kuan Lin

Soochow University, School of Computer Science and Technology,
215006 Suzhou, China
{20124227005,jxfan,20114027003,cklin}@suda.edu.cn

Abstract. In the paper, we present a polynomial-time approximation scheme (PTAS) for the minimum *k*-path connected vertex cover (M*k*PCVC) problem , which can be used to solve security problems in wireless sensor networks (WSNs), under fixed $k \geq 2$. In contrast to previously known approximation schemes for M*k*PCVC problem, our approach does not need location data of the vertices, and it can be applied to growth-bounded graphs. For any $\varepsilon_1>0$, the algorithm returns a $(1+\varepsilon_1)$-approximation M*k*PCVC. We have proved the correctness and performance of the algorithm and shown its runtime is $r \cdot n^{O(f(r))}$, where $f(r)$ is a polynomial function, $r = O((1/\varepsilon)\cdot\ln(1/\varepsilon))$ and ε is only dependent on k and ε_1.

Keywords: PTAS, k-path connected vertex cover, growth-bounded graphs, bounded degree, wireless sensor networks.

1 Introduction

It is required to ensure secure communications in most of the applications in wireless sensor networks (WSNs). However, conventional security techniques cannot be directly employed in WSNs, because sensor nodes have limited computation, power and communication capabilities, and they are usually deployed in accessible areas, where they can be easily captured or attacked [1-2]. Furthermore, it is unrealistic to make all nodes anti-tamper because of the high costs [1-2]. Hence, it is a challenge to design security protocols for WSNs.

The Canvas scheme [3-9] can provide data origin authentication in a sensor network. The *k*-generalized Canvas scheme [6] improves the Canvas scheme and ensures data integrity in the communication graph as long as at least one node on each path of the length *k*-1 is not captured. There should be two different kinds of sensor devices – protected and unprotected, and it is more difficult to capture or attack a protected one. Hence, when deploying and initializing a sensor network, it is necessary to put at least one protected node on each path of the length *k*-1 [6]. Thus, we can reduce the costs by minimizing the number of protected nodes [6], which can be abstracted as follows:

[*] Corresponding author.

We call P a k-path when the path P contains k vertices. Given a graph $G = (V, E)$ and a fixed integer $k \geq 2$. Let C be a subset of V, we call C a k-path vertex cover if each k-path in G contains at least one vertex in C. The minimum connected k-path vertex cover (MkPCVC) problem asks to find the minimum cardinality of a connected k-path vertex cover in G [1-2].

2 Related Work

MkPCVC problem has been studied a lot. Boštjan et al. [1] proved that the minimum k-path vertex cover (MkPVC) problem is NP-complete for any fixed integer $k \geq 2$. Then Tu and Zhou [10] gave a 2-approximation minimum 3-path vertex cover. Liu et al. [2] pointed out that MkPCVC problem is NP-complete and gave a (1+ε)-approximation MkPCVC for any fixed integer $k \geq 2$. Liu et al. [2] needed location information of all the vertices and repeatedly used grid-based separation and shifting strategy to get the final result. However, approaches based on shifting strategy are inherently central and can not be efficiently adapted to distributed works [11].

In the absence of position information, Nieberg et al. [12] first proposed a polynomial-time approximation scheme (PTAS) for the minimum dominating set problem in unit disk graphs. Later Kuhn et al. [11] gave local approximation schemes for the maximum independent set problem and the minimum dominating set problem in growth-bounded graphs. Then Gafeller et al. [13], Gao et al. [14] and Liu et al. [15] improved the algorithm in [12] respectively. Gafeller et al. [13] gave a (1+ε)-approximation minimum connected dominating set. Gao et al. [14] provided a (1+$\bar{\varepsilon}$)-approximation minimum d-hop connected dominating set, where $\bar{\varepsilon}$ is only dependent on d, ε and f. Liu et al. [15] separately proposed PTASs for the minimum vertex cover, weighted vertex cover and connected vertex cover problems.

In this paper, we give a (1+ε_1)-approximation MkPCVC under the constraint of bounded degree for any $\varepsilon_1>0$ based on the algorithms in [11-15].

The rest of the paper is organized as follows. In Section 3, we give some preliminaries which will be needed later. In Section 4, we present the algorithm and give the proof of its correctness, time complexity and performance. Finally, the conclusions and future works are drawn in Section 5.

3 Preliminaries

In this Section, we first provide some definitions. Then, we give several related results.

For a given graph $G = (V, E)$, if the maximum vertex degree Δ in G is upper bounded by a constant, then G is under the constraint of bounded degree [16]. Let S be the subset of V, then the subgraph of G induced by S, denoted by $G[S]$, is the graph with vertices in S and edges with both ends in S [15].

Definition 1. [12] *For any vertex $v \in V(G)$, we define $N(v)$ as the set of the vertex v and its adjacent neighbors, and $N_r(v)$ as the set of v and its r-hop neighbors (nodes that reach v via at most r-1 intermediate nodes). For any set $S \subseteq V(G)$, let $N_r(S) = \{N_r(v) | v \in S\}$.*

Definition 2. [6] *For any set $C \subseteq V(G)$, we call C a k-path vertex cover of G if each k-path in G contains at least one vertex in C. Moreover, if the graph $G[C]$ is connected, then we call C a k-path connected vertex cover. For any set $S \subseteq V(G)$, MkPCVC of $G[S]$ (see Section I) is denoted by $C_k(S)$.*

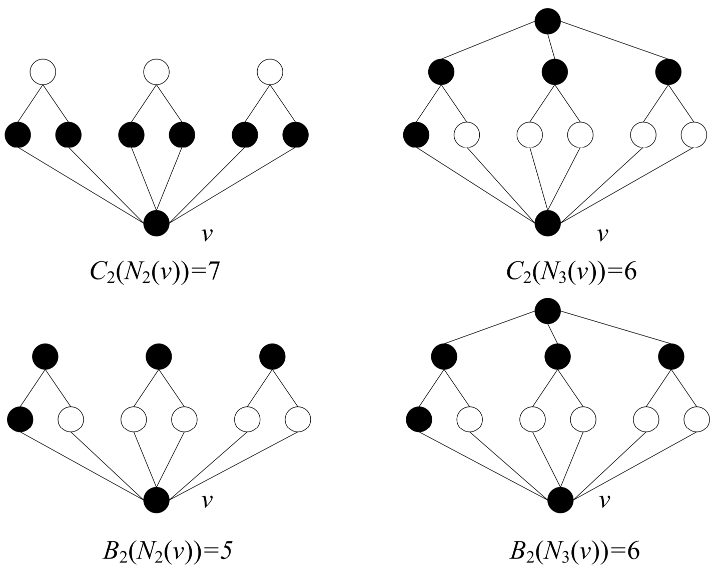


Fig. 1. An example to illustrate the difference of C and B

Liu et al. [15] used $\left|C_2\left(N_{r_i+1}(v_i)\right) \setminus N_{r_i}(v_i)\right| \le \left|C_2\left(N_{r_i+1}(v_i)\right)\right| - \left|C_2\left(N_{r_i}(v_i)\right)\right|$ to prove the performance of his algorithm. In fact, his proof is not rigorous, especially for the k-path connected vertex cover problem. For example, $\left|C_2\left(N_3(v)\right) \setminus N_2(v)\right| > \left|C_2\left(N_3(v)\right)\right| - \left|C_2\left(N_2(v)\right)\right|$ as shown in Figure 1. Therefore, we first present the concept of k-path bounded connected vertex cover in Definition 3 to solve this problem.

Definition 3. *For any set $B \subseteq N_r(v)$, we call B a k-path bounded connected vertex cover of $G[N_r(v)]$ if B is a k-path vertex cover of $G[N_r(v)]$ and each connected component in $G[B]$ contains at least one vertex in $N_r(v) \setminus N_{r-1}(v)$. For any set $S \subseteq V(G)$, the minimum k-path bounded connected vertex cover (MkPBCVC) of $G[S]$ is denoted by $B_k(S)$.*

Since $B_k\left(N_{r_i+1}(v_i)\right)\cap N_{r_i}(v_i)$ is one *k*-path bounded connected vertex cover of $G\left[N_{r_i}(v_i)\right]$, $B_k\left(N_{r_i}(v_i)\right)\le B_k\left(N_{r_i+1}(v_i)\right)\cap N_{r_i}(v_i)$. Then we have

$$\left|B_k\left(N_{r_i+1}(v_i)\right)\setminus N_{r_i}(v_i)\right|\le\left|B_k\left(N_{r_i+1}(v_i)\right)\right|-\left|B_k\left(N_{r_i}(v_i)\right)\right|. \tag{1}$$

Definition 4. *For any set $I\subseteq V(G)$, we call I an independent set if every two vertices in I are not adjacent to each other. Moreover, we call I a maximal independent set (MIS) if $I\cup\{u\}$ is no longer an independent set, for any vertex $u\in V(G)\setminus I$. For any set $S\subseteq V(G)$, a MIS of $G[S]$ is denoted by $I(S)$.*

Clearly, the subset $C\subseteq V(G)$ is a vertex cover of G if and only if $V(G)\setminus C$ is an independent set. Hence we can get a vertex cover after we find a MIS of G. Similarly, we can also get a *k*-path vertex cover after we find a *k*-hop MIS (*k*-MIS) which is defined in Definition 5.

Definition 5. [14] *For any set $I_k\subseteq V(G)$, we call I_k a k-hop independent set if there is no k-path in* $G[I_k]$. *Moreover, we call I_k a k-MIS if $I_k\cup\{u\}$ is no longer a k-hop independent set for any vertex $u\in V(G)\setminus I_k$. For any set $S\subseteq V(G)$, a k-MIS of $G[S]$ is denoted by $I_k(S)$.*

Definition 6. [12] *For any two vertices u and v in G, $dist(u, v)$ denotes the length of the shortest path from u to v in G. For any two subsets S_1 and S_2 of $V(G)$, $dist(S_1, S_2) = min\{dist(u, v)|u\in S_1, v\in S_2\}$. If $S_1\cap S_2\neq\phi$, then $dist(S_1, S_2) = 0$.*

Obviously, for any two subgraphs $G[S_1]$ and $G[S_2]$ of G, if $dist(S_1, S_2) = k$, we can add k -1 vertices to connect $G[S_1]$ with $G[S_2]$.

Definition 7. [11] *We call a graph G a growth-bounded graph if there exists a polynomial function $f(r)$ such that for every $v\in V(G)$ and $r\ge 1$, the size of the largest independent set in $N_r(v)$ is at most $f(r)$.*

Growth-bounded graphs generalize different classes of graphs including unit disk graphs, unit ball graphs and coverage area graphs. In the remainder of this section, we give two lemmas which are used in the following sections in our analysis. Lemma 1 can be derived from [15], which is used in the proof of Lemma 5. Lemma 2 , used in both Lemma 8 and Lemma 10, was applied in [2].

Lemma 1. [15] *Given a growth-bounded graph G with polynomial function f and maximum vertex degree Δ, for any vertex $v\in V(G)$ and $r\ge 1$, it holds that* $\left|B_k\left(N_r(v)\right)\right|\le(1+\Delta)\cdot f(r)$.

Lemma 2. [2] *Given a k-path vertex cover of G denoted by $C(V)$, if $G[C(V)]$ is not connected, then there exists two connected components $G[S_1]$ and $G[S_2]$ in $G[C(V)]$ such that $dist(S_1, S_2) \le k$.*

4 A PTAS for M*k*PCVC Problem

Our work is based on [11-15], which have given approximate algorithms for optimization problems in growth-bounded graphs. The main idea can be described as follows: First, the growth-bounded graph G is divided into several disjoint clusters with different radius. Then, they compute the optimal solution set of each cluster and merge all the sets into one, which is their final solution. We use the same partition scheme to compute M*k*PCVC in the growth-bounded graph G. The difference is that we use a new criterion to determine the radius of each cluster so that the problem of (1) can be solved. In addition, we let the clusters overlap more to ensure the connectivity of our solution.

Algorithm 1 PTAS for MkPCVC Problem
Input: a growth-bounded graph G with bounded degree constraint, initial parameters k and $\varepsilon > 0$
Output: a k-path connected vertex cover of G

```
1 $C \leftarrow \Phi$, $V_C \leftarrow V$;
2 { For analysis: $i = 1$; }
3 while $V_C$ is not empty do
4     choose a vertex $v$ from $V_C$;
5     $r \leftarrow 0$;
6     find the smallest radius $r$ such that $\left|B_k\left(N_{r+2k}(v)\right)\right| \le (1+\varepsilon)\cdot\left|B_k\left(N_r(v)\right)\right|$;
7     $C = C \cup C_k\left(N_{r+2k}(v)\right)$;
8     $V_c = V_c \setminus N_r(v)$;
9 { For analysis: $v_i = v, r_i = r, S_i = N_r(v), T_i = N_{r+2k}(v), i = i + 1$; }
10 end while
```

We now analyze the performance of Algorithm 1. First, we prove that the radius r_i has an upper bound to show that Algorithm 1 can be done in polynomial time by Lemma 3. Then, Lemma 4 and Lemma 5 prove that the final solution C is a k-path connected vertex cover of G. Theorem 1 provides the runtime of our algorithm. Next, we analyze its approximation performance. For each cluster S_i, Lemma 6 gives the relation between $|C_k(S_i)|$ and $|B_k(S_i)|$. Lemma 7 and Lemma 8 provide the relation between $|B_k(S_i)|$ and $|C^* \cap S_i|$ in two different conditions, and it is not considered in [13-15] in condition that S_i meets the bounds of the clusters which have been eliminated before (see in Figure 3). Then Lemma 9 gets the relation between $\sum |B_k(S_i)|$ and $|C^*|$ based on Lemma 7 and Lemma 8. In the end, Theorem 2 computes the relation between $|C|$ and $|C^*|$ (i.e., the performance of Algorithm 1) based on Lemma 6 and Lemma 9.

Liu et al. [15] has proved the radius $r \le r(\varepsilon)$ for vertex cover, weighted vertex cover and connected vertex cover problems. Next we use a similar strategy to prove the correctness of Lemma 3.

Lemma 3. *For given growth-bounded graph G with bounded degree constraint* Δ, *the radius* r_i *has an upper bound* $r(f, \varepsilon)$ *such that* $\left|B_k\left(N_{r+2k}(v_i)\right)\right| \le (1+\varepsilon)\cdot\left|B_k\left(N_r(v_i)\right)\right|$, *where f is a polynomial function of G.*

Proof. Assume that there is no upper bound of r_i, i.e., $\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| > (1+\varepsilon)\left|B_k\left(N_{r_i}(v_i)\right)\right|$ for any positive integer $r_i \ge 1$.

If $r_i = 2k\cdot t$ where t is a positive integer, then

$\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| > (1+\varepsilon)\left|B_k\left(N_{r_i}(v_i)\right)\right| > \ldots > (1+\varepsilon)^{\frac{r_i}{2k}+1}\left|B_k\left(N_0(v_i)\right)\right| \ge (1+\varepsilon)^{t+1}$;

if $r_i = 2k\cdot t+1$, then

$\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| > (1+\varepsilon)\left|B_k\left(N_{r_i}(v_i)\right)\right| > \ldots > (1+\varepsilon)^{\frac{r_i+2k-1}{2k}}\left|B_k\left(N_1(v_i)\right)\right| \ge (1+\varepsilon)^{t+1}$;

if $r_i = 2k\cdot t+2$, then

$\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| > (1+\varepsilon)\left|B_k\left(N_{r_i}(v_i)\right)\right| > \ldots > (1+\varepsilon)^{\frac{r_i+2k-2}{2k}}\left|B_k\left(N_2(v_i)\right)\right| \ge (1+\varepsilon)^{t+1}$;

...

if $r_i = 2k\cdot t+2k-1$, then

$\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| > (1+\varepsilon)\left|B_k\left(N_{r_i}(v_i)\right)\right| > \ldots > (1+\varepsilon)^{\frac{r_i+1}{2k}}\left|B_k\left(N_{2k-1}(v_i)\right)\right| \ge (1+\varepsilon)^{t+1}$.

Since $\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| \le (1+\Delta)\cdot f(r_i+2k)$ by Lemma 1, we have

$$(1+\Delta)\cdot f(r_i+2k) \ge \left|B_k\left(N_{r_i+2k}(v_i)\right)\right| \ge (1+\varepsilon)^{t+1} \ge (1+\varepsilon)^{\frac{r_i}{2k}} \tag{2}$$

for any positive integer $r_i \ge 1$. In fact, (2) has to be violated when r_i is large enough, contradicting to our assumption that $\left|B_k\left(N_{r_i+2k}(v_i)\right)\right| > (1+\varepsilon)\left|B_k\left(N_{r_i}(v_i)\right)\right|$ for any positive integer $r_i \ge 1$. □

Assume that the integer m is the highest order of polynomial function $f(r)$, then $f(r+2k) \le A_1\cdot r^m$, where A_1 is a constant. Thus we have $(1+\varepsilon)^{\frac{r}{2k}} \le (1+\Delta)\cdot A_1\cdot r^m$ for (2). Let $C_2 = (1+\Delta)\cdot A_1$, then we have

$$(1+\varepsilon)^{\frac{r}{2k}} \le A_2 \cdot r^m \ . \tag{3}$$

The smaller ε is, the larger the upper bound of r is. Hence, we can consider the upper bound $r(\varepsilon)$ when $0 < \varepsilon < \frac{1}{4k(A_2+3m)} < \frac{1}{2}$. If $\varepsilon \ge \frac{1}{4k(A_2+3m)}$, then $r \le r(\varepsilon)$.

Nieberg et al. [17] has provided the upper bound of r when $m = 2$. We can aloso use the same method to prove that the upper bound of r is $O((1/\varepsilon)\cdot ln(1/\varepsilon))$ when $m \geq 0$.

Lemma 4. *For any two vertices v_i and v_j chosen in Algorithm 1, if $T_i \cap S_j \neq \phi$, then $G[C_k(T_i)]$ is connected with $G[C_k(T_j)]$.*

Proof. Assume that the vertex u_1 is contained in $\left(T_i \setminus N_{r_i+2k-1}(v_i)\right) \cap S_j$. Then, there exists one k-path e_1 in $T_i \setminus N_{r_i+k}(v_i)$ such that u_1 is one end of e_1. Since the k-path e_1 is covered by $C_k(T_i)$, there is one vertex u_2 in e_1 such that u_2 is contained in $C_k(T_i)$. Then, we have one k-path e_2 in $C_k(T_i)$ and u_2 is one end of e_2. Obviously, e_2 is also a k-path in T_j, so e_2 is covered by $C_k(T_j)$. Thus, there is one vertex u_3 in e_2 such that $u_3 \in C_k(T_i) \cap C_k(T_j)$. Hence, $C_k(T_i) \cap C_k(T_j) \neq \phi$, i.e., $G[C_k(T_i)]$ is connected with $G[C_k(T_j)]$. □

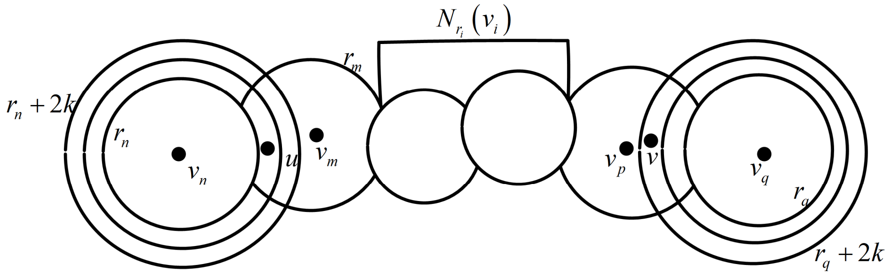


Fig. 2. This is an example to illustrate the clusters chosen in Algorithm 1

Lemma 5. *The final solution C is a k-path connected vertex cover of G.*

Proof. We first prove that C is a k-path vertex cover of G. Assume that C is not a k-path vertex cover of G. Then, there exists at least one k-path $e_u \subseteq G$ such that e_u is not covered by C. Let $e_u = (u_1, u_2, u_3, \ldots, u_k)$, then C does not contain any vertex in $\{u_1, u_2, u_3, \ldots, u_k\}$. The set V_C is empty upon completion of Algorithm 1. So, there exists a vertex v_i and a j $(1 \leq j \leq k)$ such that $u_j \in S_i$ and $\{u_1, u_2, u_3, \ldots, u_k\} \setminus u_j$ are contained in T_i. As a result, e_u must be contained in $G[(T_i)]$ and e_u is covered by $C_k(T_i) \subseteq C$, contradicting to our assumption.

Next we prove that the induced graph $G[C]$ is connected, i.e., u and v lie in the same connected component for any two vertices u and v in $G[C]$.

Assume that $u \in S_m \cap C_k(T_n)$ and $v \in S_p \cap C_k(T_q)$. As shown in Figure 2, there must be some sets S_i so that $G[S_n]$ is connected with $G[S_q]$ through $G[S_m], G\left[\bigcup S_i\right]$ and $G[S_q]$. $G[C_k(T_n)]$ is connected with $G[C_k(T_q)]$ through $G[C_k(T_m)], G\left[\bigcup C_k(T_i)\right]$

and $G[C_k(T_q)]$ can be easily derived from Lemma 4. Hence, u and v are connected in $G[C]$. (Here, S_m and S_n can be the same cluster, so can S_p and S_q.) □

Theorem 1. *The runtime of Algorithm 1 is $r \cdot n^{O(f(r))}$, where $r = O((1/\varepsilon)\cdot\ln(1/\varepsilon))$, $n = |V|$.*

Proof. The subset $C \subseteq V$ is a k-path vertex cover of G if and only if $V \setminus C$ is a k-hop independent set. If we select a k-hop independent set I_k of $G[T_i]$, then $T_i \setminus I_k$ should be a k-path vertex cover of $G[T_i]$. By Definition 7, the size of the maximal independent set of $G[T_i]$, denoted by $|\mathrm{MIS}|$, is no larger than $f(r_i)$. Since $|\mathrm{MIS}_k| \le |\mathrm{MIS}| \le f(r_i)$, we can find all the k-path vertex covers of $G[T_i]$ by exhausting search within polynomial time $n_i^{O(f(r_i))}$ where $n_i = |T_i|$, and then we can pick out $C_k(T_i)$ and $B_k(T_i)$. By Lemma 3, we know that the radius r_i has an upper bound $r(\varepsilon)$ where $r(\varepsilon) = O((1/\varepsilon)\cdot\ln(1/\varepsilon))$. Thus, the time complexity of Algorithm 1 is $\sum_i r_i \cdot n_i^{O(f(r_i))} \le r \cdot n^{O(f(r))}$. □

Lemma 6. $\left|C_k(T_i)\right| \le (1+\varepsilon_2)\left|B_k(S_i)\right|$ *where* $1+\varepsilon_2 = 1+k\cdot\varepsilon$.

Proof. By Lemma 2, for any k-path vertex cover, if we add k-1 certain vertices, the number of connected components will be reduced by one. Let d_i be the number of connected components in $B_k(T_i)$, then we can add $(d_i\text{-}1)\cdot(k\text{-}1)$ certain vertices to make $G[B_k(T_i)]$ connected. Thus we have

$$\left|C_k(T_i)\right| \le \left|B_k(T_i)\right| + (d_i - 1)(k-1) \ . \tag{4}$$

According to Definition 3, we have

$$d_i \le \left|B_k(T_i) \bigcap \left(T_i \setminus N_{r_i+2k-1}(v_i)\right)\right| \le \left|B_k(T_i) \bigcap (T_i \setminus S_i)\right| \ . \tag{5}$$

Since $B_k(T_i) \cap S_i$ is a k-path bounded connected vertex cover of $G[S_i]$ and $B_k(S_i)$ is the one with minimum cardinality, $\left|B_k(S_i)\right| \le \left|B_k(T_i) \cap S_i\right|$.

As $\left|B_k(T_i) \cap (T_i \setminus S_i)\right| = \left|B_k(T_i)\right| - \left|B_k(T_i) \cap S_i\right|$, we have

$$\left|B_k(T_i) \cap (T_i \setminus S_i)\right| \le \left|B_k(T_i)\right| - \left|B_k(S_i)\right| \ . \tag{6}$$

By (5), (6) and (7), we have

$$\left|C_k(T_i)\right| \le \left|B_k(T_i)\right| + \left(\left|B_k(T_i)\right| - \left|B_k(S_i)\right|\right)\cdot(k-1) \ . \tag{8}$$

Furthermore, we have

$$\left|B_k(T_i)\right| \le (1+\varepsilon)\left|B_k(S_i)\right| \ . \tag{9}$$

By (8) and (9), we have $\left|C_k(T_i)\right| \le (1+k\cdot\varepsilon)\left|B_k(S_i)\right|$. □

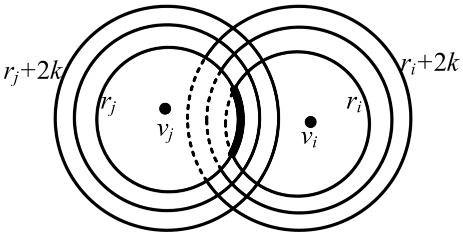


Fig. 3. This is an example to illustrate the clusters S_i and S_j chosen in Algorithm 1

Lemma 7. *For given graph G, if* $N_{r_i-1}(v_i)$ *and* $G \setminus S_i$ *are connected completely through the vertices in* $S_i \setminus N_{r_i-1}(v_i)$*, i.e., S_i doesn't meet the bound of other partitions during its construction in Algorithm 1, then* $|B_k(S_i)| \le |C^* \cap S_i|$*, where* C^* *is the optimum k-path connected vertex cover of G.*

Proof. Clearly, $C^* \cap S_i$ is a k-path vertex cover of S_i. Since $G[C^*]$ is a connected graph, both $C^* \cap N_{r_i-1}(v_i)$ and $C^* \cap (G \setminus S_i)$ are connected in $G[C^*]$. And both $C^* \cap N_{r_i-1}(v_i)$ and $C^* \cap (G \setminus S_i)$ are connected through the vertices in $C^* \cap (S_i \setminus N_{r_i-1}(v_i))$, i.e., each connected component in $G[C^* \cap S_i]$ contains at least one vertex in $S_i \setminus N_{r_i-1}(v_i)$, because $N_{r_i-1}(v_i)$ and $G \setminus S_i$ are connected through the vertices in $S_i \setminus N_{r_i-1}(v_i)$. By Definition 3, $C^* \cap S_i$ is a k-path bounded connected vertex cover of $G[S_i]$. Since $B_k(S_i)$ is the one with minimum cardinality, $|B_k(S_i)| \le |C^* \cap S_i|$. □

In step 8 in Algorithm 1, every time one partition will be eliminated from G, such as S_j. Thus, the *r-hop* neighbors of v_i in G, $N'_{r_i}(v_i)$, may be overlapped with S_j which has been eliminated before. As a result, in the partition computed in Algorithm 1, S_i may be different from $N'_{r_i}(v_i)$. As shown in Figure 3, $N_{r_i-1}(v_i)$ and $G \setminus S_i$ are connected through the vertices in $S_i \setminus N_{r_i-1}(v_i)$ and $N_{r_j+1}(v_j) \setminus S_j$ (see the heavier line in Figure 3). Then we have Lemma 8.

Lemma 8. *If* $N_{r_i-1}(v_i)$ *and* $G \setminus S_i$ *are connected through the vertices in* $S_i \setminus N_{r_i-1}(v_i)$ *and* $N_{r_j+1}(v_j) \setminus S_j$ *(see the heavier line in Figure 3, denoted by N_j), i.e., S_i meets the bound of S_j during its construction in Algorithm 1, then*
$|B_k(S_i)| \le |C^* \cap S_i| + (k-1)^2 |B_k(T_j) \cap S_i|$.

Proof. By Lemma 7, $C^*\cap S_i$ is a *k*-path vertex cover of S_i. Since $N_{r_i-1}(v_i)$ and $G \setminus S_i$ are connected through the vertices in $N_{r_i}(v_i)\setminus N_{r_i-1}(v_i)$ and N_j, each connected component in $C^*\cap S_i$ contains at least one vertex in $N_{r_i}(v_i)\setminus N_{r_i-1}(v_i)$ or N_j. Next, we need to modify $C^*\cap S_i$ to compare $|B_k(S_i)|$ with $|C^*\cap S_i|$.

After Algorithm 2, C' has $t+|Y|$ connected components connected to N_j, and the other connected components are connected to $N_{r_i}(v_i)\setminus N_{r_i-1}(v_i)$. Moreover, we have $t+|Y|\le t+|Z|\le (k-1)\left|B_k(T_j)\cap S_i\right|$. Similar to the proof of (5), we can add at most $(k-1)^2\left|B_k(T_j)\cap S_i\right|$ vertices to make C' to be a *k*-path bounded connected vertex cover of S_i. Therefore, we have

$$\left|B_k(S_i)\right|\le\left|C'\right|+(k-1)^2\left|B_k(T_j)\cap S_i\right|\le\left|C^*\cap S_i\right|+(k-1)^2\left|B_k(T_j)\cap S_i\right|.$$ □

Algorithm 2 Scheme to modify $C^*\cap S_i$
Input: $C^*\cap S_i$, a vertex *v* not contained in $C^*\cap S_i$
Output: C'(a *k*-path vertex cover of S_i with $t+|Y|$ connected components connected to N_j, and $|C'|\le|C^*\cap S_i|$)

```
1 $C' \leftarrow C^*\cap S_i$, $Y\leftarrow\{v\}$, $Z\leftarrow\Phi$;
2 while $|Z| < |Y|$ do
3   $C'=C'\setminus Y, C'=C'\cup Z$;
4   $t\leftarrow 0$, $Y\leftarrow\Phi$, $Z\leftarrow\Phi$;
5   for those components which contain vertices in $N_j$
6     if the size of the component is larger than or equal to $k$
        then $t\leftarrow t+1$ and there exists a vertex in this component which is
         contained in $B_k(T_j)\cap S_i$;
7     else
8       then let the vertex in this component with minimum degree and not
        contained in $N_j$ be $y$, and put $y$ into the set $Y$(if there is only one vertex
        contained in $N_j$ in this component, then put this vertex into the set $Y$);
        for those $k$-paths only covered by $y$ in $G[S_i]$, there exists one
        vertex, denoted by $z$, contained in $B_k(T_j)\cap S_i$ in each $k$-path. By
        Lemma 3.2, if we add $k$-2 certain vertices, the vertex $z$ can be
        connected to another connected component in $C^*\cap S_i$. Put all these
        vertices into the set $Z$;
9 end for
10 end while
```

Lemma 8 proves the conclusion under the condition that S_i meets the bound of only one partition S_j during its construction in Algorithm 1. If S_i meets more than one partition, then we have Lemma 9 and the proof is similar.

Lemma 9. *Let* $S_1' = \bigcup_{i=1}^{j} S_i$, $T_1' = \bigcup_{i=1}^{j} T_i$, $B_k(T_1') = \bigcup_{i=1}^{j} B_k(T_i)$, $S_2' = \bigcup_{i=j+1}^{q} S_i$, $B_k(S_2') = \bigcup_{i=j+1}^{q} B_k(S_i)$. *If S_{j+1}, …, S_q are the partitions which only meet the bound of S_1' during its construction, then*
$\left|B_k(S_2')\right| \le \left|C^* \cap S_2'\right| + (k-1)^2 \left|B_k(T_1') \cap S_2'\right|$.

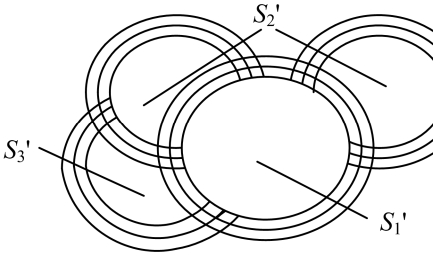


Fig. 4. This is an example to illustrate S_1', S_2' and S_3'

Lemma 10. $\sum_i \left|B_k(S_i)\right| \le (1+\varepsilon_3)\left|C^*\right|$ *holds in Algorithm 1, where* $1+\varepsilon_3 = \dfrac{1}{1-(k-1)^2\varepsilon}$ *and C^* is the optimum k-path connected vertex cover of G.*

Proof. As shown in Figure 4, denote the partitions which don't meet the bound of other partitions during its construction in Algorithm 1 by S_1, S_2, …, S_j and let $S_1' = \bigcup_{i=1}^{j} S_i$, $T_1' = \bigcup_{i=1}^{j} T_i$, $B_k(S_1') = \bigcup_{i=1}^{j} B_k(S_i)$, and $B_k(T_1') = \bigcup_{i=1}^{j} B_k(T_i)$. Denote the partitions which only meet the bound of S_1' during its construction by S_{j+1}, …, S_q and let $S_2' = \bigcup_{i=j+1}^{q} S_i$, $T_2' = \bigcup_{i=j+1}^{q} T_i$, $B_k(S_2') = \bigcup_{i=j+1}^{q} B_k(S_i)$ and $B_k(T_2') = \bigcup_{i=j+1}^{q} B_k(T_i)$. Denote the partitions which only meet the bound of S_1' and S_2' during its construction by S_{q+1}, …, S_m and let $S_3' = \bigcup_{i=q+1}^{m} S_i$, $T_3' = \bigcup_{i=q+1}^{m} T_i$,

$B_k(S_3') = \bigcup_{i=q+1}^{m} B_k(S_i)$, $B_k(T_3') = \bigcup_{i=q+1}^{m} B_k(T_i)$; … By analogy, assume that the final groups are S_n' and T_n'. By Lemma 7, we have $|B_k(S_1')| \le |C^* \cap S_1'|$. By Lemma 9, we have

$$|B_k(S_2')| \le |C^* \cap S_2'| + (k-1)^2 |B_k(T_1') \cap S_2'|;$$

$$|B_k(S_3')| \le |C^* \cap S_3'| + (k-1)^2 |(B_k(T_1') \cup B_k(T_2')) \cap S_3'|;$$

…

$$|B_k(S_i')| \le |C^* \cap S_i'| + (k-1)^2 \cdot |(B_k(T_1') \cup B_k(T_2') \cup \ldots \cup B_k(T_{i-1}')) \cap S_i'|;$$

…

$$|B_k(S_n')| \le |C^* \cap S_n'| + (k-1)^2 \cdot |(B_k(T_1') \cup B_k(T_2') \cup \ldots \cup B_k(T_{n-1}')) \cap S_n'|.$$

Thus,

$$\begin{aligned}
&|B_k(S_1')| + |B_k(S_2')| + \ldots + |B_k(S_n')| \le (|C^* \cap S_1'| + \ldots + |C^* \cap S_n'|) \\
&+ (k-1)^2 (|B_k(T_1') \cap S_2'| + \ldots + |(B_k(T_1') \cup \ldots \cup B_k(T_{n-1}')) \cap S_n'|) \\
&\le (|C^* \cap S_1'| + \ldots + |C^* \cap S_n'|) \\
&+ (k-1)^2 (|B_k(T_1') - B_k(S_1')| + \ldots + |B_k(T_{n-1}') - B_k(S_{n-1}')|) \\
&\le (|C^* \cap S_1'| + \ldots + |C^* \cap S_n'|) + (k-1)^2 \varepsilon (|B_k(S_1')| + \ldots + |B_k(S_n')|).
\end{aligned}$$

Hence, $\sum_i |B_k(S_i)| \le \sum_i |B_k(S_i')| \le \frac{1}{1-(k-1)^2 \varepsilon} \cdot \sum_{i=1}^{n} |C^* \cap S_i'| \le (1+\varepsilon_3)|C^*|$. □

Theorem 2. *For given graph G,* $|C| \le (1+\varepsilon_1)\cdot|C^*|$ *holds in Algorithm 1, where* $1+\varepsilon_1 = (1+\varepsilon_2)\cdot(1+\varepsilon_3)$ *and* C^* *is the optimum k-path connected vertex cover of G.*

Proof. $\sum_i |C_k(T_i)| \le (1+\varepsilon_2) \cdot \sum_i |B_k(S_i)|$ by Lemma 6, and $\sum_i |B_k(S_i)| \le (1+\varepsilon_3) \cdot |C^*|$ by Lemma 10. Therefore, we have $\sum_i |C_k(T_i)| \le (1+\varepsilon_2) \cdot (1+\varepsilon_3) \cdot |C^*|$. □

5 Conclusion

In the paper, we present a PTAS for M*k*PCVC problem, which can be applied in designing security protocols for WSNs [6], in growth-bounded graphs under the constraint of bounded degree. In contrast to [2], our algorithm only relies on the adjacent relation between vertices and doesn't need the geometric representation of the vertices. Also, our algorithm can be easily extended to a distributed one based on a greedy method. However, there may only be one active cluster in the graph in each round, which leads to a linear time complexity [11]. Hence, improving our algorithm into a totally distributed one and applying it to WSNs will be our future work.

Acknowledgments. This work is supported by National Natural Science Foundation of China (No. 61170021), Application Foundation Research of Suzhou of China (No.SYG201240).

References

1. Brešar, B., Kardoš, F., Katrenič, J., Semanišin, G.: Minimum k-path vertex cover. Dis. Appl. Math. 159, 1189–1195 (2011)
2. Liu, X., Lu, H., Wang, W., Wu, W.: PTAS for the minimum *k*-path connected vertex cover problem in unit disk graphs. J. Global Opt. 56, 449–458 (2013)
3. Gollmann, D.: Protocol analysis for concrete environments. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2005. LNCS, vol. 3643, pp. 365–372. Springer, Heidelberg (2005)
4. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (2010)
5. Novotný, M.: Formal analysis of security protocols for wireless sensor networks. Tatra Mt. Math. Publ. 47, 81–97 (2010)
6. Novotný, M.: Design and analysis of a generalized canvas protocol. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 106–121. Springer, Heidelberg (2010)
7. Vogt, H.: Integrity preservation for communication in sensor networks. Tech. Rep. 434, ETH Zurich, Institute for Pervasive Computing (2004)
8. Vogt, H.: Exploring message authentication in sensor networks. In: 1st European Workshop Security Ad-Hoc Sensor Networks (2004)
9. Vogt, H.: Increasing Attack Resiliency of Wireless Ad Hoc and Sensor Networks. In: 2nd International Workshop on Security in Distributed Computing Systems, pp. 179-184 (2005)
10. Tu, J., Zhou, W.: A factor 2 approximation algorithm for the vertex cover P3 problem. Info Proc. Lett. 111, 683–686 (2011)
11. Kuhn, F., Moscibroda, T., Nieberg, T., Wattenhofer, R.: Local approximation schemes for ad hoc and sensor networks. In: DIALM-POMC Cologne, Germany, pp. 97-103 (2005)
12. Nieberg, T., Hurink, J.: A PTAS for the minimum dominating set problem in unit disk graphs. In: Approximation and Online Algorithms, pp. 296-306 (2006)
13. Gfeller, B., Vicari, E.: A Faster Distributed Approximation Scheme for the connected Dominating Set Problem for Growth-Bounded Graphs. In: 6th International Conference on Ad-Hoc, Mobile, and Wireless Networks, pp. 59-73 (2007)
14. Gao, X., Wang, W., Zhang, Z., Zhu, S., Wu, W.: A PTAS for minimum *d*-hop connected dominating set in growth-bounded graphs. Opt Lett 4, 321–333 (2010)
15. Liu, Y., Fan, J., Wang, D., Du, H., Zhang, S., Lv, J.: Approximation Algorithms for Vertex Cover Problems in WSN Topology Design. Ad Hoc and Sensor Wireless Networks (accepted)
16. Wang, Z., Wang, W., Kim, J.M., Thuraisingham, B., Wu, W.: PTAS for the minimum weighted dominating set in growth bounded graphs. J. Global Opt. 54, 641–648 (2012)
17. Nieberg, T., Hurink, J.L., Kern, W.: A robust ptas for maximum weight independent sets in unit disk graphs. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 214–221. Springer, Heidelberg (2004)

A Simple and Effective Long Duration Contact-Based Utility Metric for Mobile Opportunistic Networking

Chyouhwa Chen, Wei-Chung Teng, and Yu-Ren Wu

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei, Taiwan
{similar,weichung}@csie.ntust.edu.tw,
M10015076@mail.ntust.edu.tw

Abstract. Mobile devices with Bluetooth or WiFi modules may form an infrastructure-less network, called mobile opportunistic network (MON), through their opportunistic contact with one another. The contact pattern between the mobile devices (or nodes) reflects the human users who carry them. Therefore end-to-end paths between the mobile devices in the network do not exist almost all the time. Messages must be carried by the mobile nodes and forwarded one hop at a time when two nodes are in contact. The sparseness of the contact patterns makes the data dissemination problem on MON a difficult one. In this paper, we identify the essential role of long-duration contacts and propose an effective routing scheme based on long-duration contacts. Our proposal involves minimal state maintenance overhead and is easy to implement. Extensive simulations are conducted in comparison to several approaches using real world mobile contact traces and the results support the effectiveness of our approach.

Keywords: Mobile opportunistic network, long-duration contacts, BubbleRap.

1 Introduction

Today's smart phones and many mobile devices, for example cameras, carried by human users are equipped with integrated Bluetooth or WiFi modules. These devices may connect with each other when the human users move into each other's transmission range. The connectivity of these mobile devices form a natural infrastructure-less mobile opportunistic networks (MON), or delay tolerant networks (DTNs) [1]. However, the contact pattern between the mobile devices reflects the human users who carry them and is characterized by the sparseness of the connectivity between these devices in the sense that the end-to-end paths between most of the mobile devices in the network may not exist almost all of the time [1]. The sparseness of the contact patterns means that messages can be forwarded one hop at a time, only when two nodes are in contact, which makes data dissemination problem on MONs difficult. However, due to the explosive growth of the mobile devices and potentially unlimited novel applications, the problem has attracted much attention in the networking research community in recent years [1], [17], [21].

Message routing, or how to choose the next mobile node in contact that is the most conducive to successful delivery of a message to its destination, is at the center of the data dissemination problem. Numerous message routing schemes have been proposed to differentiate nodes that are more likely to deliver content or bring it closer to the destination [1], [17], [21]. They differ in what knowledge is extracted from the past observed contacts between nodes and how to use it as a basis to predict future contact opportunities. The *utility-based* routing schemes are a particularly simple and easy to implement class of algorithms. A utility-based scheme assesses the strength of ties between nodes based on easy to measure metrics. Metrics proposed in the literature include time of last encounter, contact frequency, and degree centrality, and so on [1], to identify nodes that are highly mobile or social. Though many different utility-based schemes have been proposed, no previous approaches differentiate between short and long duration contacts as use them as the basis in the design of utility functions. In [7], it was hinted that short duration contacts may not be essential for effective message diffusion. In this paper, we investigate this intuition systematically and propose a simple yet effective algorithm to show that long duration contacts indeed do have better predictive power, and therefore should be used as the main basis when message forwarding decisions are being made.

We use a simple experiment to evaluate the predictive power of long-duration contacts. For a random pair of (source, destination) nodes, we calculate the empirical probability $p(D/X=n)$ of a source encountering a destination in the future after having had n number of long-duration encounters with the destination. The results are shown in Fig. 1, which also includes the results when only short duration contacts are taken into account. It is clear that nodes have a higher probability of meeting with a destination in the future when it has a higher probability of long-duration encounters with it in the past.

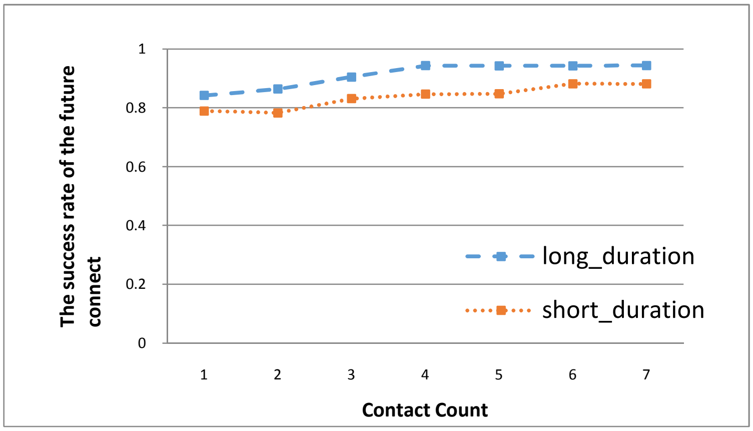


Fig. 1. Empirical prediction capacity of long duration contacts in Cambridge trace

Based on the observations discussed above, we investigate and propose a utility-based algorithm that is easy to implement and has better or competitive message delivery performance at much lower delivery costs when compared to other utility-based approaches and a much more elaborate community detection scheme [8]. The fact that our approach significantly outperforms the other schemes in terms of the cost of message delivery translates into conserved energy for the devices, which may be the most important objective in a mobile networking environment. We summarize the key contributions of our work as follows:

- We propose the design of a simple long duration contact-based (LDC) utility function as the basis for message forwarding decision.
- Based on the utility function, we propose a message forwarding algorithm that is simple to implement in a distributed manner.
- We evaluate our proposal extensively using real world contact traces and demonstrate the performance advantage of our proposed scheme against other related schemes for the information dissemination problem.

The remainder of this paper is organized as follows: we review the related works in Section II. We then present the details of our approach in Section III. The simulation and evaluation are shown in Section IV. Finally, we conclude our work in Section V.

2 Related Work

Much work has been done to understand the encounter patterns between the mobile devices (or nodes) or humans [7], [9]. Observations from large empirical traces show that the encounter patterns exhibit a number of unique characteristics. For example, most mobile nodes have low percentage of unique encounters among the whole population [7]. Since end-to-end connections almost never exist between any two nodes in the network, messages delivery must be done in a store-and-forward manner where messages are carried by the mobile nodes and forwarded one hop at a time when two nodes come into contact. The data dissemination problem is a more difficult form of message routing problem than that in, for example, traditional ad hoc networks. To make intelligent routing decisions, knowledge must be extracted from the observed contacts as a basis for future contact prediction. The past observed contacts between nodes are usually summarized into an aggregate network graph [6], with graph edges representing (one or more) past encounters between the vertex nodes. An edge in this graph conveys how strong the tie between two nodes are based on, for example, information about often two nodes encounter each other. The origination of these encounters may be either because nodes have a strong social relation (friends), or because they co-locate accidentally without actually knowing each other (familiar strangers).

Facing the unique challenges in routing on MONs, many proposals have been made in the literature to combat the challenges for data dissemination. We classify them into three broad categories: utility-based approaches, community detection-based approaches, and real social feature-based approaches [1], [2], [5], [8], [12], [15], [16], [17], [18], [21]. In the first approach, the usefulness of a node for delivering messages to destinations is assessed using a utility function. When a message holding node contact another node, the utility function is calculated, then the node having higher utility value for the message's destination is given the message. Early approaches make use of simple measures in the design of utility functions, such as time of last encounter, node contact frequency, and so on [12]. Later efforts make use of concepts from complex network analysis, such as degree centrality, node betweenness, and node similarity combined with betweenness [11]. Early work on SimBet is a famous example in this approach [2]. In SimBet, the utility of a relay node n for a destination D depends on both its betweenness, and an assessment of the degree of similarity between the contacted nodes and that of D. In the community detection-based approaches, it was observed that it may be fruitful to explicitly extract the relatively stable human community structures from past node contacts and use them as the basis for relay node selection [8]. A distributed community detection algorithm may be used to extract the community structures from the contact traces. Message routing algorithms are designed by exploiting the extracted communities as a basis. A seminal proposal in this approach is BubbleRap [8], [16], [17]. The routing of a message proceeds in two stages. In the first stage, the goal is to deliver it to the destination's community as quickly as possible. In the second stage, inter-community contacts are preferred so the message can reach the destination quickly. In recent years, a number of researchers have attempted to exploit the real, or declared, social features (such as interests, affiliations, etc.) of the mobile nodes, which is the most stable relationship between nodes, as a basis for message forwarding [15], [17], [18], [21]. A number of message routing approaches have been proposed to exploit the social features information available. For example, some approaches extend the betweenness measure in the social feature setting to by preferring popular (in terms of social features) relay nodes, while other algorithms adopt the approach of preferring relay nodes with high degree of similarity in social features [17].

3 A Simple Long-Duration-Contact-Based Information Dissemination

In this paper, we investigate two important facets in node contact patterns to measure the tie strength and use them as the basis for effective routing algorithm design: *number of long duration contacts*, and *dissimilarity between contacts*. The existence of any long duration contacts between nodes is regarded as an indicator of substantive strength in the relationship between them. However, it is known strangers, or Vagabonds, that represent the rest of the population must be utilized for effective message forwarding[4], [19]. This fact is illustrated in Fig. 2, which shows the successful message delivery ratio performance using the Cambridge trace. The curve marked by "LDC-only" is the one where messages are only exchanged between nodes with long duration contacts between them. Clearly the success ratio of the LDC-only approach is lower than that of the BubbleRap strategy. Therefore, how to make use of the Vagabonds is essential.

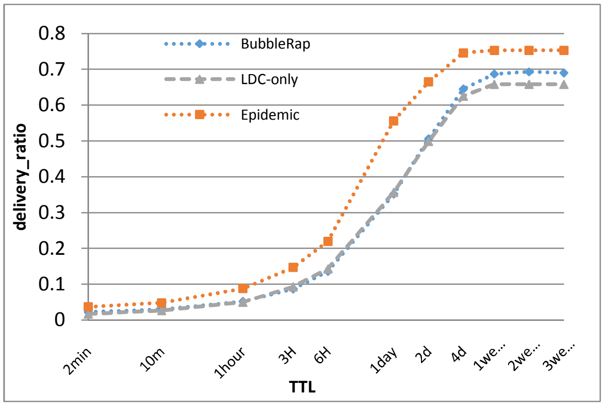


Fig. 2. Delivery ratio performance comparison when only long duration contacts are used for message forwarding using Cambridge data set

Based on the observations, we design a utility function for relay node selection, which comprises of two components: the number of long duration contacts and a measure of dissimilarity between nodes as an indicator for Vagabond node identification. The notations are explained in the following table. When node n_i meets a potential relay node n_j, n_i will have contacts that is common to n_j, denoted by $Sim(i,j)= N(i)\cap N(j)$, and those that n_j has never met before, denoted by $Dissim(i,j) = N(i)\text{-}Sim(i,j)$.

Table 1. Notations used in this paper

Notation	Description
$LDC_i(d)$	Number of long duration contacts between n_i and d, where d is the destination of a message
$N(n_i)$	Contacts of n_i in the aggregated network graph so far
$Sim(i,j) = N(i)\cap N(j)$	Contacts common to both n_i and n_j
$Dissim(i,j) = N(i) - Sim(i,j)$	The degree to which node n_i is a Vagabond relative to n_j

The utility function we propose is shown in eq(1), which has two components. The first component gives a larger utility value when a relay node has many long duration contacts with the destination and thus will have a high probability of being in the same community with the destination. The second component is designed to allow a limited preference to a relay node if it is a stranger, i.e. Vagabonds relative to the current message holder. The utility value of a relay node is a linear combination of its degree of tie strength with the destination node and its degree of being Vagabonds.

$$U_{ni}(d) = \alpha * LDC_i(d)/(LDC_i(d)+LDC_j(d)) + \beta * Dissim(i,j) \,/\, (Sim(i,j)+Dissim(i,j))$$

$$U_{ni}(d) = \alpha * LDC_i(d)/(LDC_i(d)+LDC_j(d)) + \beta * Dissim(j,i) \,/\, (Sim(i,j)+Dissim(j,i)) \qquad (1)$$

Based on the utility function, Fig. 3 depicts our proposed message forwarding algorithm. When a message holder n_i, with M number of copies of a message, encounters a node n_j without that message, there are two cases to consider. First, if either node has had contacts with the message destination node, n_i computes the utility function value for both nodes, and forwards M/2 copies of the message to n_j if n_j has a higher utility value. Secondly, if neither node has encountered the destination node before, and n_i has sufficiently large number of copies of the message and considers n_j to be sufficiently different, it will also forward M/2 copies of the message. The condition for unfamiliarity is defined to be n_i having met n_j for a very small number of times (< 3 in the experiments).

```
when  node $n_i$ becomes in contact with node $n_j$  then
  for each message with M number of copies carried by $n_i$ do
    if either node $n_i$ or node $n_j$ has had contacts with destination of M  then
      if  $Un_i(d) < Un_j(d)$  then
        Send M/2 to node $n_j$
    else
      if  $n_i$ is sufficiently unfamiliar with $n_j$ and $n_i$ has enough # of message
      copies then
        Send M to node $n_j$
```

Fig. 3. Long Duration Contact-based Message Forwarding Algorithm

4 Simulation and Evaluation

We evaluate the performance of the proposed LDC-based message forwarding algorithm through trace driven simulations. We have compared our proposed scheme with many other forwarding schemes in the literature. We present the performance comparison results between our scheme, the epidemic scheme, and the BubbleRap scheme only, based on two popular datasets in the literature: MIT reality mining trace [3] and Cambridge trace [14]. For each simulation run, we generated 5000 messages from a random source node to a random destination node at each *t* seconds. The range of parameter values and the default parameter values are shown in the following table. The nodes are assumed to have enough buffer space for all messages, and the bandwidth is high to allow the exchange of all messages between nodes, similar to the assumptions used in previous studies [6], [12], [13].

Table 2. Parameters used in the simulation

Parameter	Values	Default
Number of initial copies of each unique message	2, 4, 6, 8	2
TTL	2min, 10min, 1hr, 3hr, …	N/A
Weight parameter α in the utility function	1, 0.8, 0.6, …	0.8
Long duration threshold value --Cambridge	300, 500, 700, 900 (sec)	500
Long duration threshold values -- MIT	50, 250, 450, 650 (sec)	200

4.1 Message Delivery Ratio Performance

Fig. 4 shows the message delivery ratio with the Cambridge data set as a function of TTL. Obviously Epidemic has the highest delivery ratio among all forwarding algorithms because of greedy forwarding behavior. Our proposed strategy LDC outperforms BubbleRap across all TTL value for both data sets. Comparing Fig. 4 and Fig. 2, the improvement in performance is quite significant when our utility function for is used. The results again confirm the importance of Vagabonds for effective message forwarding. It is surprising that our simple utility-based scheme outperforms the much more elaborate BubbleRap scheme.

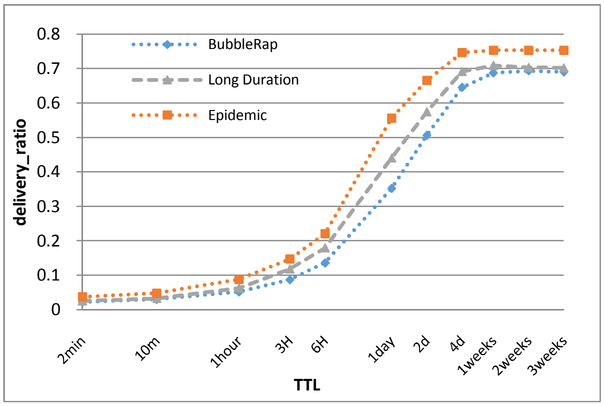


Fig. 4. Delivery Ratio Performance of LDC for Cambridge trace

4.2 Delivery Cost Performance

Fig. 5 shows the delivery cost performance results. Epidemic incurs significantly higher overhead than the other algorithms due to its nature of message flooding. LDC always incurs the lowest delivery cost for both traces. For the Cambridge data set, LDC incurs 40% less delivery overhead than BubbleRap. Even more significantly, for the MIT trace, LDC incurs around 80% less delivery cost than BubbleRap when the TTL value is longer. Clearly, our proposed scheme is extremely effective in selecting good relay nodes out of all contacts for message delivery.

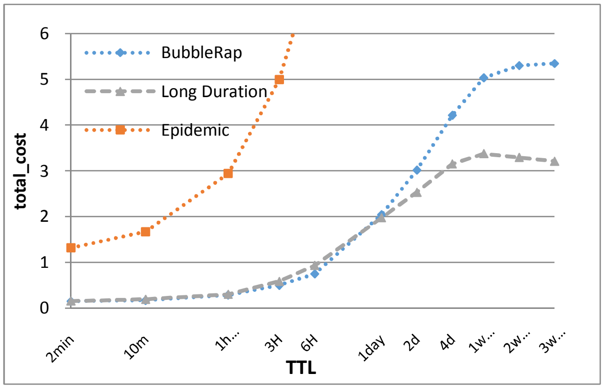


Fig. 5. (a) Delivery Cost Performance of LDC for Cambridge trace

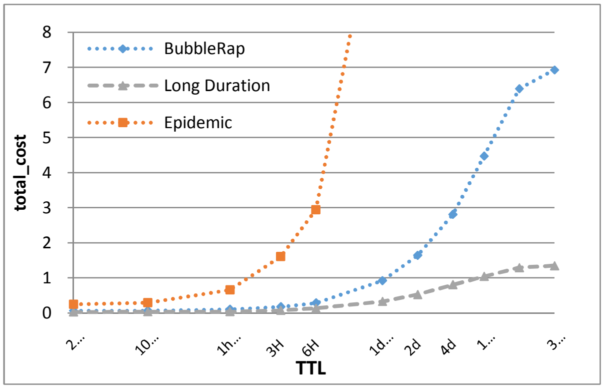


Fig. 5. (b) Delivery Cost Performance of LDC for MIT trace

4.3 Impact of Weight Parameter Values in Utility Function

The results are shown in Fig. 6. The parameter values of $\alpha = 0.8, \beta = 0.2$ seem to be a good choice for the Cambridge data set and for other data sets in general, indicating the consistency in the relative importance between community nodes and Vagabond nodes across a wide selection of data sets.

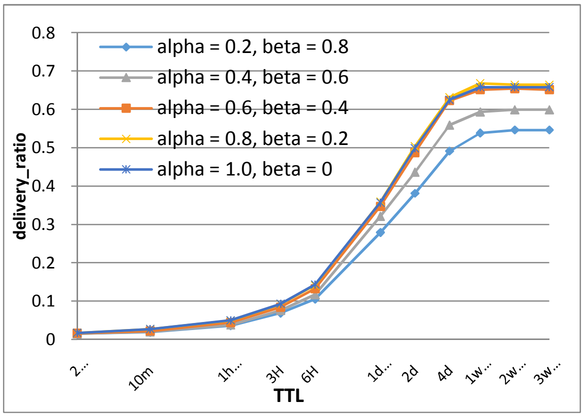


Fig. 6. Performance Impact of weight parameter values for Cambridge trace

4.4 Impact of Long Duration Threshold Values

The results of the impact of the long duration threshold values on the Cambridge data set are shown in Fig. 7. It is apparent different threshold values do not have much impact on system performance. The results indicate that our design is robust with respect to the duration threshold parameter. More importantly, the results may reflect the fact that long duration contacts are truly indicative of the community structure between nodes. Therefore different values do not have much impact on the system performance. Thus, instead of employing the distributed community detection method like that in BubbleRap, it is fruitful to pay special attention to those contacts that are of longer durations.

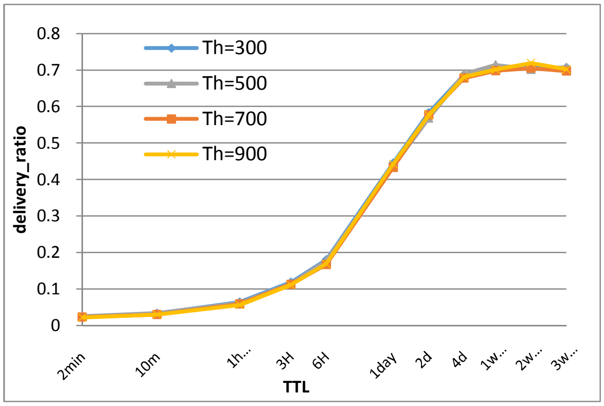


Fig. 7. Performance Impact of threshold values for the Cambridge trace

5 Conclusion

We have presented and evaluated our LDC-based scheme with two real world traces. The results show that the proposed scheme has better or at least competitive message delivery performance, and significantly better delivery cost performance compared to the much more elaborate BubbleRap scheme. As for our future work, we plan to study the performance of our algorithm using more real world traces that represent different network settings to validate the effectiveness of our approach. Second, we intend to investigate the issues surrounding automatic determination of optimal duration thresholds. Finally, as the utility function is neutral to architectural design of routing algorithms, we intend to extend the utility function to detected community approaches, such as BubbleRap, and real social feature-based approaches, such as ML-SOR [15] and others [18], to enhance their performance in mobile opportunistic networks.

References

1. Cao, Y., Sun, Z.: Routing in Delay/Disruption Tolerant Networks: A Taxonomy, Survey and Challenges. IEEE Communications Surveys and Tutorials 15 (2013)
2. Daly, E.M., Haahr, M.: Social Network Analysis for Information Flow in Disconnected Delay-Tolerant MANETs. IEEE Transactions on Mobile Computing 8 (2009)
3. Eagle, N., Pentland, A.: CRAWDAD data set mit/reality (v. July 01, 2005), `http://crawdad.org/mit/reality/`
4. Gaito, S., Pagani, E., Rossi, G.P.: Strangers Help Friends to Communicate in Opportunistic Networks. Computer Networks 55, 374–385 (2011)
5. Gao, W., Cao, G.: User-Centric Data Dissemination in Disruption Tolerant Networks. In: 30th Annual IEEE International Conference on Computer Communications, INFOCOM (2011)
6. Hossmann, T., Spyropoulos, T., Legendre, F.: Know Thy Neighbor: Towards Optimal Mapping of Contacts to Social Graphs for DTN Routing. In: 29th Annual IEEE International Conference on Computer Communications, INFOCOM (2010)
7. Hsu, W., Helmy, A.: On Nodal Encounter Patterns in Wireless LAN Traces. IEEE Transactions on Mobile Computing 9 (2010)
8. Hui, P., Crowcroft, J., Yoneki, E.: BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks. IEEE Transactions on Mobile Computing 10 (2011)
9. Kim, J., Helmy, A.: The Evolution of WLAN User Mobility and Its Effect on Prediction. In: 7th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 226-231 (2011)
10. Meroni, P., Gaito, S., Pagani, E., Rossi, G.P.: CRAWDAD Data Set unimi/pmtr (v. December 01, 2008), `http://crawdad.org/unimi/pmtr/`
11. Newman, M.E.J.: The Structure and Function of Complex Networks. SIAM Review 45 (2003)
12. Pagani, E., Rossi, G.P.: Utility-based Forwarding: A Comparison in Different Mobility Scenarios. In: 3rd International ACM Workshop on Mobile Opportunistic Networks, MobiOpp (2012)

13. Pietilänen, A.-K., Diot, C.: Dissemination in Opportunistic Social Networks: the Role of Temporal Communities. In: 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 165–174 (2012)
14. Scott, J., Gass, R., Crowcroft, J., Hui, P., Diot, C., Chaintreau, A.: CRAWDAD data set cambridge/haggle/imote/cambridge (v. January 31, 2006), `http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/cambridge`
15. Socievole, A., Yoneki, E., Rango, F.D., Crowcroft, J.: Opportunistic Message Routing using Multi-layer Social Networks. In: 2nd ACM Workshop on High Performance Mobile Opportunistic Systems (2013)
16. Wei, K., Zeng, D., Guo, S., Xu, K.: On Social Delay-Tolerant Networking: Aggregation, Tie Detection, and Routing. IEEE Transactions on Parallel and Distributed Systems 25, 1563–1573 (2014)
17. Wei, K., Liang, X., Xu, K.: A Survey of Social-Aware Routing Protocols in Delay Tolerant Networks: Applications, Taxonomy and Design-Related Issues. IEEE Communications Surveys and Tutorials 16, 556–578 (2014)
18. Wu, J., Wang, Y.: Social Feature-based Multi-path Routing in Delay Tolerant Networks. In: 31st Annual IEEE International Conference on Computer Communications, INFOCOM (2012)
19. Zyba, G., Voelker, G.M., Ioannidis, S.: Diot: Dissemination in Opportunistic Mobile Adhoc Networks: The Power of the Crowd. In: 30th Annual IEEE International Conference on Computer Communications, INFOCOM (2011)
20. Bui Xuan, B., Ferreira, A., Jarry, A.: Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks. International Journal of Foundations of Computer Science 14 (2003)
21. Zhu, Y., Xu, B., Shi, X., Wang, Y.: A Survey of Social-based Routing in Delay Tolerant Networks: Positive and Negative Social Effects. IEEE Communications Surveys and Tutorials 15, 387–401 (2013)

Adaptive QoS and Security for Video Transmission over Wireless Networks: A Cognitive-Based Approach

Walid Abdallah[1], Suk kyu Lee[2], Hwagnam Kim[2], and Noureddine Boudriga[1]

[1] Communication Networks and Security Research Lab. University of Carthage, Tunisia
[2] Wireless and wired Inter-Networking Research Lab, Korea University, South Korea
{ab.walid,noure.boudriga2}@gmail.com,
{sklee25,hnkim}@korea.ac.kr

Abstract. Transmitting video streams while ensuring security is a challenging issue in wireless networks. Due to the openness of the wireless medium, encryption techniques should be used in many critical applications to secure video stream transmission. However, this may impact the quality of service required to efficiently reconstruct video data caused by the processing overhead and delays generated by the encryption procedure. In addition, a degradation of the transmission performance in terms of signal to noise ratio (SNR) and bit error rate can occur due to the varying characteristics of the wireless environment, which affect the available bandwidth on the channel. Thus, adaptable approach that can make a trade-off between security enforcement and transmission performance is required. In this work, we investigate the design of a video stream transmission scheme over wireless networks that can ensure security and efficiency. Our proposal is cognitive-based in the sens that security policy is selected according to the risk level evaluated by the receiver. This is done using real-time assessment of the network security state, intrusion detection, access history, and resource availability in the mobile device. The main goal is to satisfy the real-time constraints of the video stream and guarantee clear reconstitution of specific part of the scene qualified as the region of interest (ROI) by adapting the compressing parameter, qualified as the delivery ratio, according to the selected security policy and the transmission characteristics of the wireless channel.

Keywords: Wireless networks, video transmission, security, cognitive.

1 Introduction

Wireless networks have known a tremendous development during the last years. Many novel applications are being implemented exploiting the advances achieved in smart phone technology and improvements in wireless communication. Particularly, transmitting multimedia content over wireless networks is envisioned in many critical applications, such as video surveillance, videoconferencing, and telemedicine. A major issue in these applications is ensuring communication security whilst satisfying the real-time constraints of the video stream. The openness of the wireless medium, the dynamic nature of the network topology, the user mobility, and the multi-hop routing, are the main factors which facilitate many security attacks including eavesdropping, jamming, black holes, and denial of service. These malicious attacks can threaten the privacy of the transmitted data and disturb normal operation of the network. As a countermeasure, security sensitive applications require the use of encryption primitives to ensure security for multimedia data in terms of confidentiality, integrity, and authentication. Nevertheless, this may affect the performance of the video transmission due to its sensitivity to Quality of Service (QoS) parameters such as delay, bandwidth, and error rate. Hence, satisfying QoS constraints of video stream in wireless mobile environment is a challenging issue especially when cryptography must be employed to protect information. Applying encryption may generate extensive computation and communication overheads and add an extra delay to video streaming transmission impacting its decoding and reconstruction efficiency. Due to the unpredictably of the wireless channel, a trade-off between security and quality of service constraints must be achieved taking into account the currently available resources and the risk level encountered by data transmission.

Most of works in the literature consider adaptation of either security or QoS and not both. Very few researches had been conducted to resolve the antagonist nature of security and QoS in wireless mobile networks[2,9,8,4,5]. The majority of these works try to adapt particular security parameters (encryption algorithm, key sizes, ...) to QoS constraints of the transmitted data. They do not consider particular context and characteristics of the wireless channel when selecting the optimized security policy. In addition, these models do not handle specific information gathered and learned from the environment or take into account the quality of experience observed by the mobile on the discovered wireless networks.

In this paper, we study the development of an adaptive and secure transmission system for video streams in wireless mobile environment. Our design is cognitive-based where the receiving party will decide about the security policy and the transmission parameters according to the observed security state of the access network, the available bandwidth on the wireless channel, and resource occupancy state in the mobile device. The selected parameters will be communicated to the transmitter in order to be enforced on the video stream transmission process. In this context, appropriate compression and encryption functionalities should be selected. Compression is highly required in wireless communication to reduce the large size of multimedia data and enable its transfer over the limited bandwidth of the wireless channel. To this end, many compression algorithms have been proposed. The conventional approach consists in applying compression on every image without analyzing which region is more important than others. Although this method can be very effective in reducing the size of data, it does not consider the need of the user to have better viewing quality of some regions in the image. This is very useful in many applications including video surveillance, medical imaging analysis, and video conferencing. Thus in this work we will use the concept of region of interest (ROI) developed in [7]. In this approach, a user defined region of the image will be transmitted without any compression; however the remaining parts will be compressed according to a delivery ratio defined in function of the wireless link state reflected by the Signal to Interference and Noise Ratio (SINR). In this paper, a model for determining the delivery ratio had been elaborated taking also into consideration the used processing power and the generated communication overhead caused by selecting suitable encryption algorithms with specific key size. The main contributions of this paper are as follows:

– The design of an adaptive scheme to secure video stream transmission over wireless mobile networks. The proposed scheme enables mutual authentication between the transmitter and the receiver and secure key exchange using elliptic curve cryptography. The security policy is selected according to the assessed risk and trust level evaluated using cognitive approach.
– The development of an empirical model that adapts the delivery ratio parameter according to the selected cryptography primitives and the state of the wireless channel in terms of SINR.
– The implementation of a set of encryption schemes that can be selected to enforce the security policy on video stream transmission.

The remaining parts of this papers are as follows: Section 2 assesses the related work; Section 3 describes the proposed cognitive secure video transmission scheme; Section 4 discusses the implementation of the scheme and some performance evaluation work. Section 5 concludes the paper.

2 Related Work

Achieving adaptable transmission of video streaming over wireless networks is considered as one of the main issues in the networking research community. The objective is to meet the quality of service performances that allow an efficient video playback regardless the varying characteristics of the wireless channel. This problem becomes harder when security measures are needed to protect the content of the video stream due to the delay generated by encryption. Several works [2,9,8,5,10] had tried to make a trade-off between both security and QoS by adaptively selecting appropriate configurations that ensure acceptable security level while limiting the impact of encryption on resource usage and transmission performances. Reza et al.[9] investigated security and QoS for video transmission over mobile ad-hoc networks (MANETs). They proposed a QoS aware adaptive security (QaAS) scheme to overcome the problem of the delay incurred by the application of encryption algorithms on the video stream. This scheme adapts the encryption primitives according to the required multimedia proprieties even if this can lead to the degradation of the security level. This work selects security primitives that can perform the encryption operation within a fixed period taken as a threshold which depends on the delay constraints of the video sequence. It does not consider any other criteria for making adaptation such as the security risk level observed on the network, the available bandwidth, or the measured transmission latency and jitter.

Another adaptive security architecture was proposed by Ben Mahmoud et al.[8] to ensure reliable communication in future aircraft. The design of a secure system for the embedded network was performed based on a central component named Security Manager. The work in [4] studies the features of applying adaptive encryption on video streaming over 3G mobile networks. The security scheme is based on the SCTP transport protocol. The authors use this technique to implement sensitive applications such as medical video stream transmission. Alia et al.[2] addressed the problem of considering simultaneously security and QoS. He proposed an adaptive model based on the selection of component structures that can make dynamic and fine-grained trade-offs between security and QoS. This model is multi-constraints and utility-based that takes into account resources availability, risk level, and user preferences. These works address the problem of secure transmission of video data over wireless channel using adaptive approaches. They do not implement any cognitive perception that can be used to decide about the required security level in a given circumstance. Moreover, authors do not consider the wireless link state and the available transmission capacity in selecting the security and multimedia encoding parameters.

In this paper, we investigate the development of a cognitive-based security architecture for video transmission over wireless mobile networks where the security risk level and the transmission characteristics of the wireless media are taken into consideration when selecting appropriate security policy and encoding parameters. The cognitive network is able to perceive the current state, plan, decide, and apply particular actions[3]. In addition, it can learn from the environment and make future decisions to achieve an overall common goal.

A first problem is related to the large volume of data generated by video sources which overload the wireless channel capacity. Therefore, specific video encoding techniques that can compress video data are required to cope with the limited bandwidth of the wireless channel. The standard H264/MPEG4[1] is one of the most used digital video compression techniques which employs the discrete cosine transform (DCT) and inter-frame time dependency to reduce the data size of the video stream. This paper adopts the encoding technique presented in [7] that is also based on the DCT to achieve compression. However, the processing executed after applying the DCT is different. In MPEG, a Huffman coding based quantization is performed on the whole image to reduce the size of the data that will impact its entire content. However, the proposed technique is based on selecting a specific zone in the image referred to as the region of interest (ROI). Due to the importance of the ROI it will be transmitted without applying any compression. On the other hand, the region of non interest (Non-ROI) will be transmitted according to a delivery ratio that depends on the current wireless link state. Hence, the available bandwidth will be mainly used to transmit pixels in the ROI with a maximum quality and the remaining capacity is used to transmit the Non-ROI with less quality by eliminating some pixels.

3 Secure Video Transmission Architecture

In this section, we describe the architecture of the proposed scheme. We consider a mobile device that roams in a given area and discovers many heterogeneous access networks with divers technologies (GPRS, UMTS, Wi-Fi) and characteristics (Bandwidth, security policy,..), that can be used to connect and receive video streams. Therefore, the mobile device must select the most appropriate network that can satisfy the QoS and security constraints of the media stream service. Our proposal is cognitive-based where the receiver will assess the security level of the network and the characteristics of the wireless channel in terms of signal to interference and noise ratio (SNIR). Then, he will select a security policy where encryption and authentication methods with specific key sizes are chosen according to the required security level. According to the cryptographic primitives and the determined capacity of the wireless channel, a compression rate will be selected to encode the video stream. The selected security policy and the compression rate are sent to the transmitter to settle its transmission and security parameters in a real-time fashion. The architecture of the proposed video transmission scheme is depicted by Figure 1. In the sequel, we will describe the different modules which encompass the proposed scheme.

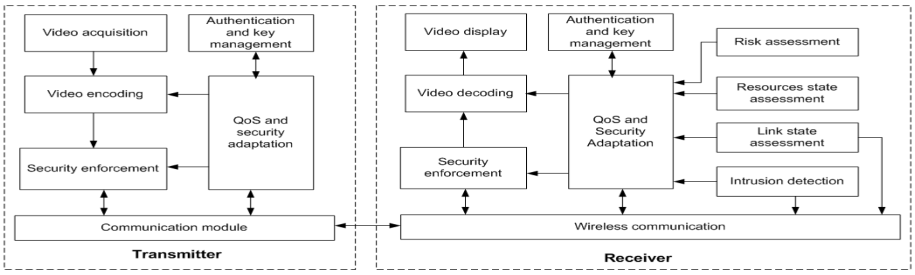


Fig. 1. System Architecture

3.1 Cognitive Perception Functionalities

The perception functionalities are used to assess the characteristics of the environment in terms of security and transmission performance.

Risk Assessment. The risk assessment module evaluates the level of security threats when the mobile devices is connected to a specific wireless access network. The output is a probability that is estimated according to specific criteria. However, this operation should be performed in a short period of time to satisfy the real-time requirement of the video stream and to be executed by the mobile devices with low computation power. In this work, we choose a very simple risk assessment method based on three characteristics. The first is related to security of accessing to the wireless network which can be open access or secure access

Table 1. Risk Classification

Type of access	Nature of the network	History of access	P_r
Open	Public	New	0.9
Open	Public	Familiar	0.8
Open	Private	New	0.7
Open	Private	Familiar	0.6
Secure	Public	New	0.5
Secure	Public	Familiar	0.4
Secure	Private	New	0.3
Secure	Private	Familiar	0.2

using for example, the WEP protocol. The second parameter concerns the nature of the wireless network to which the mobile device is connected. In this case, we can classify it as a public network deployed for example in airports, internet cafe, or a private network that is managed by the user or by a known party. The last parameter is historical access to the network that reflects the fact that the mobile device connects for the first time to the wireless network or it usually access to it. A classification of the risk is performed according to Table 1.

Intrusion Detection. The main objective of the intrusion detection module is to identify abnormal behavior and triggers an alarm if an attack is detected. This can be done by inspecting traffic exchanged by the mobile devices with different network elements which is similar to a host intrusion detection system. In addition, we can assume that the mobile device is also able to examine traffic transmitted on the network. Nevertheless, this will require an extensive processing capability and high memory capacity which are not available in resource limited devices. To overcome this limitation we can assume that many mobile devices can cooperate together to perform the network intrusion detection. Thus, if an attack is identified by one device an alert will be sent to all mobiles in an ad-hoc fashion. According to the severity of the alert the mobile will determine an attack probability, P_a. If we suppose that there is five levels of alarm severity $L \in \{1, 2, 3, 4, 5\}$, then the attack probability can be calculated as, $P_a = 0.2\,L$. Off-course if no alarm is generated $P_a = 0$.

Resources State Assessment. The resource state assessment module gathers information about memory and CPU availability in the mobile device. The objective is to select less or more lightweight encryption primitives according to the percentage of available resources. We define a resource availability ratio denoted by P_{re}, which is calculated as follows $P_{re} = 1 - \frac{M+C}{200}$ where M is the percentage of memory occupancy and C is the percentage of CPU usage.

Link State Assessment. The link state assessment module controls the transmission performances based on measuring the signal to interference and noise ratio. The retrieved SNIR is used to evaluate the capacity of the wireless channel by calculating the maximum data bit rate that can be transmitted given by the Shannon formula $R = W\,log_2(1 + SNIR)$, Where W is the bandwidth of the channel. As it will be detailed later, this will be used to determine the compression ratio applied to encode the video stream in the transmitter.

3.2 Cognitive Decision and Adaptation Functionalities

After retrieving the environment security and transmission characteristics, the mobile device will decide about the security policy that will be applied to secure the video stream and then define the compression rate with which data will be encoded to satisfy its real-time constraint and reconstitution performance. This is performed in the QoS and security adaptation module as follows.

Table 2. Security Classes

Class	encryption		Authentication	Security ratio
	ROI	Non-ROI		
0	no	no	no	$0 \leq SR < 0.2$
1	no	no	HMAC-SHA1	$0.2 \leq SR < 0.4$
2	AES-128	no	HMAC-SHA1	$0.4 \leq SR < 0.6$
3	AES-128	AES-128	HMAC-SHA1	$0.6 \leq SR < 0.8$
4	AES-192	AES-128	HMAC-SHA1	$0.8 \leq SR$

Selection of the Security Policy. The objective of this step is to determine suitable encryption and authentication algorithms to secure video transmission according to security ratio, SR. This ratio is calculated using the risk probability P_r, attack probability P_a, and resource availability ratio P_{re}, as follows $SR = \alpha P_r + \beta P_a + \gamma P_{re}$ and $\alpha + \beta + \gamma = 1$, α,β, and γ are coefficients defined by the administrator. According to the calculated security ratio a security class will be selected. A security class is defined by a set of encryption and authentication algorithms used to secure the video stream. In this work, we used the AES algorithm with 128 bits and 196 bits key sizes to provide confidentiality. We use selective encryption to apply different security levels in the ROI and Non-ROI parts of the image. Also, a Hash Message Authentication Code (MAC) based on SHA1 ensures origin authentication. This MAC is calculated on the entire blocks of the image after applying the encryption algorithm. The enforcement of the security policy requires the establishment of secret keys. This is done, in our scheme, using elliptic curve public key encryption. Table 2 is used to map between the security ratio, the security class and cryptographic algorithms.

Compression Rate Determination. After selecting the appropriate security policy according to the security risk encountered by the transmitted data, we must define the encoding parameters of the video stream in order to ensure acceptable performance when reconstituting the scene. This will take into account the available capacity of the wireless channel and the cryptographic primitives applied to protect the transmitted data. In this work, we adopted a compression approach of the video stream based on the concept of region of interest (ROI). If we consider a video as a sequence of images that are transmitted with a given rate, a ROI is a zone in each image that is the most important for the user. In other words, it is a set of pixels in the image defined by the user and must be transmitted with maximal quality. Here, quality is defined by the delivery ratio corresponding to the fraction of pixels that are transmitted without any error. As an example in video surveillance the user should focus on a specific region in the image and not on all the image. Also, in video conferencing users are more interested in recognizing the faces of each other than the surround environment.

In this work we choose to apply a delivery ratio of 100% for the ROI pixels, however the delivery ratio of the Non-ROI pixels will depend on the security policy and the available link capacity. It is trivial to say that applying encryption will reduce the delivery ratio due to delays generated by the processing and communication overheads of the cryptographic functions. If we denote by D_i the delivery ratio of the non-ROI corresponding to the security class, i. We have $D_i = \alpha_i D_0$, where $0 < \alpha_i \leq 1$ and $\alpha_0 = 1$. The factors α_i will be determined using measurements performed after the implementation on a specific plate-from. On the other hand, we show that D_0, which corresponds to the delivery ratio where no security is applied is determined by the expression, $D_0 = \frac{\frac{R}{R_I} - ROI}{N \times n - ROI}$, where R is the available bit rate of the wireless channel, R_I is the image rate of the video stream (30 image/s), ROI is the number of bits of the region of interest, N is the number of pixels in each image, and n is the number of bits that encode each pixel.

3.3 Authentication and Key Distribution

Before establishing any secure session, the transmitter and the receiver must authenticate each other and agree on specific keys to apply encryption primitives on the video stream. Although the authentication method does not affect the QoS service parameters of video stream however it is very important to ensure the overall security of the system. In our proposal, the authentication technique will also depend on the selected security class. Two main authentication methods are defined. The first consists in using a pre-shared key (PSK) that is pre-configured in the transmitter and the receiver. This technique is not very secure due to the fact that if the PSK is discovered the system will be vulnerable to many security attacks. Nevertheless, it has the advantage to be simple and less complex. In our architecture the PSK based authentication method will be selected for security classes 1 and 2. A more secure authentication method is based on public key encryption and particularly digital signature where the transmitter and the receiver will exchange certified public keys during secure session establishment. This technique requires that both the transmitter and the receiver trust a common certification authority which makes it more costly and complex to implement. For this reason, we choose to apply this technique when classes 3 and 4 are selected to secure the video stream transmission.

Besides, the transmitter and the receiver should establish a common secret key before they can use cryptographic technique. In this context, the Diffie-Hellman key exchange procedure is one the most used techniques that is based on the intractability of the Discrete Logarithm Problem (DLP). However, its processing overhead and large key size are not appropriate for wireless and mobile devices equipped with very limited processing power and reduced memory capacity. Elliptic curve cryptography [6] can be used to cope with this issue by reducing the required key size and performing key establishment in a limited delay.

To apply the elliptic curve based key exchange procedure, the transmitter and the receiver agree on an elliptic curve space, $E(F_p)$ in the form of $E(F_p) = \{(x, y) \in F_p^2,\ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$, where$F_p$ is the Galois field constructed by a given prime number, p, two values a and$b$ satisfying $4a^3 + 27b^2 \neq 0$, and $\mathcal{O}$ the neutral element of the curve. Also, they must select a generator point, G. The receiver generates a random number $n_r < p$ and calculates the point $Y_r = n_rG$ and sends it to the transmitter. Similarly, the transmitter generates a random number n_t and calculates the point $Y_t = n_tG$ and sends it to the receiver. The established key is determined by each party by multiplying the received elliptic curve point by its generated secret number, therefore we have $K = n_rY_t = n_tY_r = n_tn_rG$. To avoid man-in-the middle attacks, the exchanged public points must be authenticated either by the PSK or using the public key digital signature technique according to the selected security class as it has been described above. Note that the key K is not directly applied to secure video stream transmission; but two different keys denoted by, K_e and K_a are derived from this key and other information such as the PSK or public key material. These keys will be used by the encryption and MAC primitives respectively.

3.4 Encoding/Decoding Function

Once the security class i has been selected and the corresponding delivery ratio D_i has been determined, these two parameters will be sent by the receiver to the transmitter to encode and encrypt the video stream. The encoding process is performed using discrete cosine transform (DCT) that is applied on the non-ROI zone using the same procedure described in [7]. To this end, the signal representing each image is decomposed into a number of macro blocks where each one is represented with, $\alpha_{uv}(x, y) = I[u\,w{+}x,\ vh{+}y]$, u and v are respectively the column index and row index of the macro block, x and y are the coordinates of the pixel, I is the original signal generated by the image, and w, h are respectively the width and the height of the macro block. The DCT is applied to generate the signal β_{uv} as:

$$\begin{aligned}\beta_{uv}(x, y) = &\tau(x)\tau(y)\sum_{n=0}^{w-1}\sum_{m=0}^{h-1}(\alpha_{uv}(n, m) \\ &cos(\frac{\pi(2n+1)x}{2w})\, cos(\frac{\pi(2m+1)y}{2h}))\end{aligned} \tag{1}$$

Where τ represents the handling function defined as:

$$\tau(k) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad k = 0 \\ \sqrt{\frac{2}{N}} & for \quad k = 1, 2, .., N-1 \end{cases} \tag{2}$$

Then, we perform the zigzag scanning order of β_{uv} to transform it into a one dimensional signal ρ_{uv}. This sequence is compressed and transformed into γ_{uv} by the following operation

$$\gamma_{uv}(k) = \begin{cases} \rho_{uv}(k) & if\ k \leq L \times D_i \\ 0 & otherwise \end{cases} \tag{3}$$

where L is the length of the macro block and D_i is the delivery ratio when the security policy of the class i is applied to secure the video stream. This transformation is applied only to macro blocks that belong to the non-ROI zone of the image. As it has been indicated previously, all blocks of the ROI will be transmitted without any reduction of the number of pixels, ie. a delivery ratio equal to 1. The next step consists in quantifying the signal γ of each block to be encoded into a finite number of bits.

The final step is encrypting the encoded signal and generate the message authentication code HMAC using the algorithms specified by the security policy. After receiving the signal, the receiver verifies the MAC and then decrypts it. Finally the original image was recovered using the inverse DCT.

3.5 Security Policy Enforcement

In this subsection, we detail the implementation of the cryptographic primitives to secure the video stream. The security enforcement process consists in applying the selected encryption and origin authentication procedures with appropriate key sizes. This process is executed after encoding the image using DCT and the defined delivery ratio of the non-ROI as previously described. The first step after encoding is to encrypt the image. In our security architecture, we adopted a stream cipher encryption technique which has the advantage of ensuring security to video stream while preserving an acceptable performances needed to efficiently recover the video sequence in the receiver. In addition, this technique is more adapted to the macro-block based encoding scheme. In our implementation, each encoded pixel of the image that must be encrypted (ROI or both ROI and Non-ROI) will be Xor-ed with a pseudo-random key that is generated by applying the AES algorithm in the OFB (Out-put feed-back) mode using the encryption key K_e and a randomly generated initial vector IV.

After applying encryption, the MAC function is executed on all blocks of the image using the authentication key K_a. This process is more efficient than calculating the MAC before encryption, because it prevents costly processing overhead due to decrypting compromised messages.

4 Performance Evaluation

In this section, we present the performance evaluation of the proposed system, which has been conducted with some application level benchmarks in the testbed. We utilized *Kinect* and *OpenNI SDK* as the primary source to acquire the images with a resolution of 640x480 pixels. The entire framework is created by *C++* in *Ubuntu 10.04*. We built up the transmitter on the hardware platform consisting of an *Intel Core i5* CPU M450 2.40GHz, 3.00 GB RAM, *GeForce* 310M. The receiver's platform was composed of *Intel Core i5-2410M* CPU 2.30GHZ, 4.00 GB RAM, *GeForce* GT 520M. The wireless settings were IEEE 802.11 b/g/n mixed. In order to evaluate the effectiveness of the system, we conducted our experiment within a mobile scenario. We started moving the receiver from the point where the signal strength was high to the area where the signal strength was low and came back to the original location. Furthermore, we evaluated the performance of the proposed system with the maximum physical transmission rates of 300 Mb/s, 150 Mb/s, and 54 Mb/s with respect to the transmission time and the peak signal-to-noise ratio (PSNR).

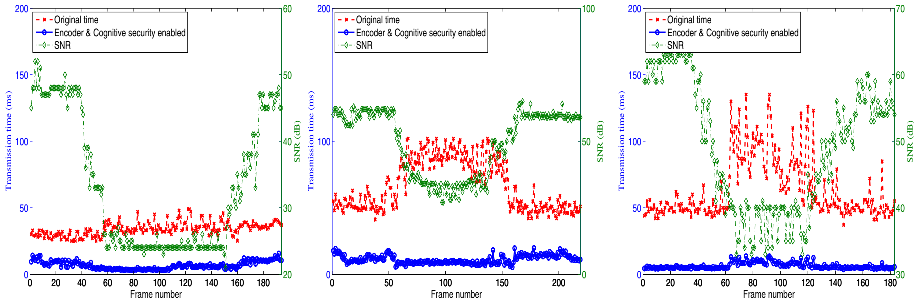
4.1 Encryption Time

For the first experiment, we conducted a measurement of the computational time needed to enforce the different security classes shown by Table 2. The average computational time was 0.1 ms, 0.84 ms, 3 ms, and 3.28 ms for Class 1, 2, 3, and 4. The computational time for Class 1 does not cost as much as we expected, since the only operation that it is required is authentication. However, the computational time for encryption process was higher than the authentication. Furthermore, as the key length size increases, the computational time increases. Based on this experiment, we were able to obtain the parameterα_i for every security class to determine the delivery ratio D_i. We found that α for each security class was 0.98, 0.95, 0.85, and 0.8 for Class 1, 2, 3, and 4. These acquired values will be used in the following experiments described in the sequel.

4.2 Evaluation in Terms of Transmission Time

In this set of tests we evaluate the performance of our proposed scheme in terms of transmission delay. We consider a video sequence composed of 180 frames that are transmitted from a fixed source to a mobile receiver. We assess delay to transmit each frame in the two cases when our scheme is applied to secure video stream and when it is not applied.

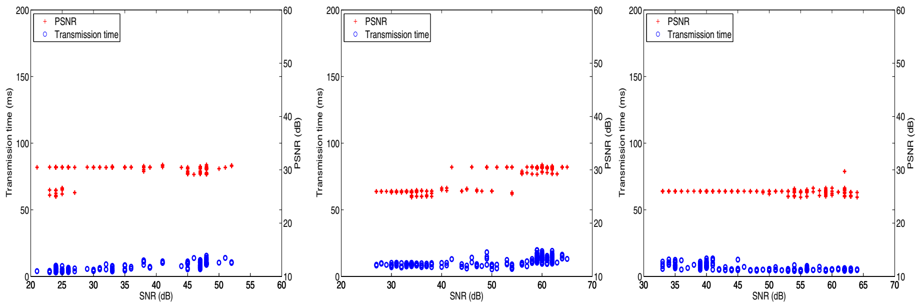
Figures 3a, 3b, and 3c compare between the transmission time for the different frames when the the proposed security scheme was applied (encoder and cognitive security enabled) and the case when no security procedure is used (original time). The security level was set to Class 1. In these figures, the left y-axis indicates the transmission time of every frame in ms, the right y-axis illustrates the SNR variation in terms of dB due to the mobility of the receiver, and the x-axis represents the sequence of video frames. The results are evaluated for three



(a) 300 Mbit/s **(b)** 150Mbit/s **(c)** 54Mbits/s

Fig. 2. Transmission time for different physical transmission rates

values of the physical transmission rate, 300 Mb/s, 150 Mb/s, and 54 Mb/s. We can see that our scheme reduces the transmission time of the frames. Also, the transmission delays of all frame are very close even the SNR of the wireless channel is varying very fast. This observation is the same for all the considered transmission rates. We can conclude that our security scheme ensures enhanced adaptivity for video stream transmission to the characteristics of the wireless channel and to the required security level.



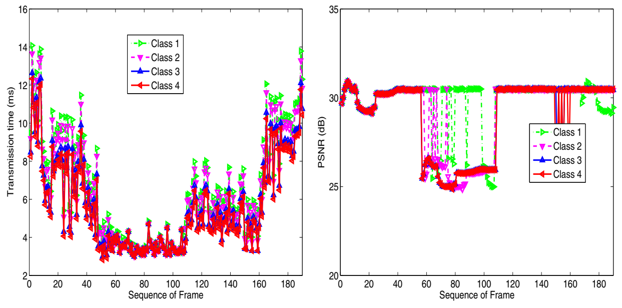
(a) 300 Mbit/s **(b)** 150Mbit/s **(c)** 54 Mbit/s

Fig. 3. Distribution of transmission time and PSNR

Along with the analysis of the transmission time, we evaluated our system in terms of image quality. This is reflected by the PSNR of the reconstituted frame. Figures 3a, 3b, and 3c show the distribution of the transmission time and the PSNR of the image. The left y-axis in these figures describes the transmission time in ms, the right y-axis indicates the PSNR in dB, and the x-axis represents the SNR starting from the range of 20 dB to 70 dB. The obtained results for the PSNR were 29.89 dB, 28.47 dB, and 25.87 dB for the physical transmission rate of 300 Mb/s, 150 Mb/s, and 54 Mb/s. Moreover, we could observe that the image quality was consistent even with different link state quality. Thus, with higher transmission rates, we could enhance the video stream quality.

4.3 Performance Evaluation for Different Security Classes

In the last set of experiments, we evaluated the proposed system with respect to the security classes. Figures 4a and 4b depict the transmission time and the PSNR when different security classes are applied for a sequence of video frames. As we can see, the transmission times for different classes are slightly close. This is due to the adaptation applied by our scheme on the delivery ratio. Indeed, as we applied the encryption process with a longer key size, the computational time for security increases. Since the delivery ratio is adjusted according to the computational overhead of the security enforcement, the delivery ratio was set lower to the case when the security enforcement did not take place. Thus, reducing the transmitted data volume will lead to decreasing the transmission time. We can observe that this contributes in decreasing the image quality reflected by the PSNR. We can see also that even increasing the required security class may lower the image quality, in the worst case this will be less than 1 dB which can be considered as an acceptable value.



(a) Transmission time **(b)** PSNR

Fig. 4. Performance comparison among different classes

5 Conclusion

In this paper, we investigated QoS and security provision for video transmission in wireless network. We proposed a secure and efficient video stream transmission scheme over wireless mobile networks based on the concept of cognitive security which consists in tailoring the security policy according to the security state of the network and resource availability in the mobile device. Our scheme allows selective encryption where the ROI of each image is encrypted with longer key size and transmitted with higher quality and the non-ROI is encrypted and compressed with adapted parameters that depend on the assessed security state of the networks and the transmission characteristics of the wireless channel. The implementation and experimental study show the effectiveness of the proposed cognitive approach in protecting the content and satisfying the real time constraints of the video stream. Considering selected frame encryption approach to enhance the adaptability of the proposed scheme to the characteristics of the wireless environment can be envisioned as a future work.

References

1. H.264: Advanced video coding for generic audiovisual services
2. Alia, M., Lacoste, M., He, R., Eliassen, F.: Putting together QoS and security in autonomic pervasive systems. In: Proceedings of 6th ACM Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWinet 2010), Bodrum, Turkey, pp. 19–28 (October 2010)
3. Haykin, S.: Cognitive Dynamic Systems: Perception-Action Cycle, Radar and Radio. Cambridge University Press, New York (2012)
4. Kamphenkel, K., Blank, M., Bauer, J., Carle, G.: Adaptive encryption for the realization of real-time transmission of sensitive medical video streams. In: International Symposium on a World of Wireless, Mobile and Multimedia Networks WoWMoM 2008, Newport Beach, CA, pp. 1–6 (June 2008)
5. Kang, K.D., Son, S.H.: Towards security and qos optimization in real-time embedded systems. ACM SIGBED Review 3(1), 29–34 (2006)
6. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
7. Lee, S.K., Ryoo, J., Yoo, S., Jung, J., Lee, W., Kim, H.: Cosa: Adaptive link-aware real-time streaming for mobile devices. In: Proceedings of IEEE WiMob 2013, Lyon, France (October 2013)
8. Mahmoud, M.S.B., Larrieu, N., Pirovano, A., Varet, A.: An adaptive security architecture for future aircraft communications. In: Proceedings of the IEEE/AIAA Digital Avionics Systems Conference (DASC 2010), Salt Lake City, UT, pp. 3.E.2-1–3.E.2-16 (October 2010)
9. Reza, T.A., Barbeau, M.: QaASs: Qos aware adaptive security scheme for video streaming in manets. Journal of Information Security and Applications 18(1), 68–82 (2013)
10. Venkatramani, C., Westerink, P., Verscheure, O., Frossard, P.: Securing media for adaptive streaming. In: Proceedings of the Eleventh ACM International Conference on Multimedia (MULTIMEDIA), New York, NY, USA, pp. 307–310 (2003)

Virtual Network Mapping Algorithm in Wireless Data Center Networks

Juan Luo[1], Wenfeng He[1], Keqin Li[1,2], and Yaling Guo[1]

[1] College of Computer Science and Electronic Engineering, Hunan University, Hunan University, Changsha, Hunan410082,China
juanluo@hnu.edu.cn

[2] Department of Computer Science, State University of New York, New Paltz, New York 12561, USA
lik@newpaltz.edu

Abstract. In a wireless data center (WDC) network, link interference and node mobility may affect resource allocation. In this paper, we propose the wireless virtual network embedding algorithm based on link interference (WVNEA-LI). We model the virtual network and the physical network in a WDC. Moreover, we allocate nodes and links simultaneously based on a concurrent mechanism. According to a link interference matrix, in the paper, we carry out dynamic adjustment of link allocation to balance the coordination of nodes and links. Simulation results show that the WVNEA-LI algorithm can improve the acceptance ratio of a virtual network, raise the benefit-cost ratio, and increase efficiency of network resources.

Keywords: Link interference, resource allocation, virtual, WDC.

1 Introduction

With the development of cloud technologies, cloud services have received wide attention, such as cloud storage, cloud platform, cloud computing, and so on. Recently, a multitude of companies have competed to build large data centers (DC) to provide hardware support for cloud services. A data center network (DCN), as one of the most important parts to build a DC, needs to bind millions of servers and provide cloud technology with optimal bandwidth. The extremely high frequency (EHF) technology (especially 60 GHz wireless communication technology) is able to employ wireless communication to transmit data with high rate (throughput up to 4 Gbit/s). As a result, it expands wireless communication technology to DCN. The 60 GHz wireless technology supports DCN effectively with low cost, high density, and rapid connection. Therefore, resource allocation is an intricate issue in wireless data centers (WDC) [1].

In addition, the network virtualization technology has received extensive attention from both industry and academia for resource allocation. A wireless network bridges the gap between many access/core networks and wireless communication technology [2]. To support different customized services, wireless network virtualization realizes to operate multiple virtual wireless networks on the same physical network. The core of network virtualization is virtual network embedding. It allocates underlying physical network resources to each virtual network request, and hence, satisfies the virtual network requests and minimizes the usage of the underlying network resources as far as possible. In cable network architecture, the main focus of network virtualization resource allocation is to map the virtual nodes and links to appropriate physical nodes and links [3,4,5], whereas link interference and node movement in WDC negatively affect the resource allocation in wireless networks. Compared with the inherent isolation of cable link, the broadcast nature of wireless links brings about interference over wireless link while assigning virtual network, thus it impacts the performance of a virtual network.

In this paper, we propose the WVNEA-LI (Wireless Virtual Network Embedding Algorithm based on Link Interference) algorithm, which considers the interference relation among links. Besides, it adopts the concurrent allocation method to map nodes and links simultaneously. The algorithm could improve the coordination and efficiency of allocation between nodes and links, and provide a promising approach to achieving virtual network allocation in wireless networks.

The rest of the paper is organized as follows. Sect. 2 depicts the most related work. Sect. 3 analyzes virtual resource allocation problems and introduces the related wireless network virtualization models. Sect. 4 describes the proposed wire-less virtual network embedding algorithm based on link interference. Sect. 5 discusses the performance evaluation of the proposed algorithm. Finally, Sect. 6 concludes the paper and presents our plan for future work.

2 Related Work

The existing network virtualization resource allocation algorithms for wireless net-works are limited. Nevertheless, there has been a lot of research on the network virtualization resource allocation [6,7]. Yun et al. introduced the virtual network allocation framework in the wireless network environment, but they did not give a specific allocation algorithm [2]. Fan and Ammar proposed a random game model based on the service provider and the foundational service provider. The model proved that, at a given forecast price, there is a Nash equilibrium during the bidding process. Furthermore, the service providers need to reflect their own price function accurately [8].

Recently, a data center network (DCN) suffers from the congestions caused by un-balanced traffic distributions [9,10]. The efficiency of a data center network is limited by oversubscription, and the typical unbalanced traffic distributions of a DCN further aggravate the problem [11].

Based on the existing 60 GHz radio frequency (RF) technology, Shin et al. [12] proposed that the fundamental limitation of wireless data centers is that the maximum number of effective connections in the network is directly proportional to the full volume occupied by the data center divided by the radiating volume of a single antenna beam. Consequently, they integrated wireless transceivers and switching logic within each server node, and allocated them in cylindric racks to establish a semi-regular mesh topology, which they called it Cayley data center. Moreover, in their Cayley model, there exited interference among links while the links are in the same launch angle of a node. The authors did not discuss the interference in the Cayley model, however.

Szeto et al. [13] introduced a virtual network mapping algorithm based on link an-ti-interference. The algorithm works as follows. (1) Set candidate allocations for allocation. (2) Check the feasibility of the candidate allocations. (3) Assess the allocation quality. (4) Select the optimal candidate allocation to map. The algorithm ensures the least link interference after virtual network allocation and considers the node and link resources constraints. However, its complexity is too high. Every allocation needs K candidate allocations and multiple comparisons, which greatly increase the time and cost of allocation.

Ahn and Yoo [14] described a wireless access network allocation algorithm for wireless mesh networks, which was designed on double antenna architecture by using orthogonal frequency division multiplexing (OFDM). In order to guarantee the coordinate channel allocation across different links while meeting the constraint of limited orthogonal channel, they applied a polynomial approximate solution and proposed an optimization algorithm based on genetic algorithms. The algorithm includes two basic algorithms: greedy algorithm and genetic algorithm based on greedy strategy. The combining greedy algorithm could get an accelerating convergence. They handled link interference with the channel allocation algorithm, and improved the overall performance of network by adopting the enhanced genetic algorithm, which all confirmed the feasibility of virtual access networks. But the allocation algorithm ignores the node and link resources constraint conditions, and only works over the wireless mesh network. In addition, it is not a distributed algorithm, which is required.

Paul and Seshan [15] proposed an online virtual wireless network allocation algorithm based on approximate KaRe graph. The algorithm includes an online scheduling method and an approximate KaRe graph mapping algorithm. The algorithm divides the wireless resources into several small rectangular units, and the requested resource is regarded as multiple rectangular units. Each rectangle unit is based on the product of the time slot and orthogonal frequency. The underlying wireless resources are divided into several zones by applying the domain division method, and the virtual network is allocated to the corner of zones or the highest EDI (distribution density index) zones as far as possible. As a result, this approach maximizes the utilization of resources through maximizing the occupied domains density and minimizing the vacant domains density. However, the algorithm never considers the spatial dimension and the combination of nodes and links.

Luo et al. [16] proposed a wireless virtual network embedding algorithm based on link reliability named WVNEA-LR. Through pretreating physical network topology and allowing multiple virtual nodes of the same virtual request mapping to the same physical node, the algorithm improves the acceptance ratio of virtual networks and saves the physical link resources. According to Q factor, WVNEA-LR over-comes the weakness that a sparse topology may reduce the acceptance ratio of virtual networks. Lv et al. [17] proposed a novel POC-enabled channel allocation algorithm under OFDMA-based WMN wireless mesh network (WMN) architecture they designed to exploit partially-overlapped channels and support multiple virtual access networks simultaneously. But both the two algorithms hardly consider link interference.

3 System Model

In a wireless network, underlying physical nodes have the computing and storage resources, and physical links have the bandwidth resources. A two-step mapping algorithm is widely adopted in the current virtual resource allocation algorithms. It works as follows: (1) allocate all virtual nodes; (2) allocate the virtual links after all the virtual nodes are allocated. Thus, it enables node allocation to be separated from link allocation which affects coordination. Whereas, a two-step mapping algorithm, which adopts the single objective optimization scheme in the two stages, may lead to problems. On one hand, a two-step mapping algorithm takes the recall method while it finds that it cannot satisfy all constraints of the link demand after all nodes are mapped, resulting in low allocation efficiency. On the other hand, it is difficult to get the global optimal allocation of the virtual network. In addition, in wireless networks, the packet loss rate of underlying physical links is significantly higher than that of a wired environment. The reason is that the inherent natural isolation of wire links guarantees less interference between links, while a wireless link has the inherent broadcast feature, which brings about severe interference.

The factors on wireless link reliability are the mobility of wireless nodes, environment, interference of the wireless links allocation, and so on. Among these factors, the interference between wireless links impacts resources allocation primarily. The selection and allocation of the current wireless link could trigger interference on allocated virtual network. Under the condition of serious interference, the network service that has been allocated might be interrupted.

3.1 Wireless Virtual Request

Due to the variability of wireless link bandwidth, the allocation and computation of a virtual network becomes particularly complicated in wireless network environment. We assume that the link bandwidth of the underlying physical network is fixed and the link with bandwidth bearing is under the constraint of channel. Every virtual network request dynamically comes, stays, and then leaves after completing service.

We describe a virtual network as an undirected weighted graph $G^V = (N^V, L^V)$, where N^V is the node set of virtual network requests, L^V is the link set of virtual network requests. Each virtual node $n^v \in N^V$ contains the property $CPU(n^v)$, which indicates the computing ability. Each virtual link $l^v \in L^V$

contains the property $bandwidth(l^v)$, which is the bandwidth requirement of virtual link l^v. *VnRequest* is shown in Fig. 1(a). For example, the CPU of virtual node c is 25, and the bandwidth of the virtual link between node a and node c is 20.

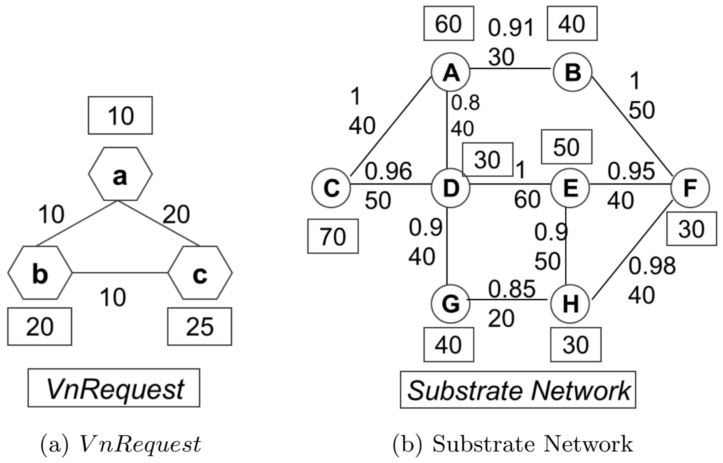


Fig. 1. Wireless virtual request (*VnRequest*) and Physical network (substrate network)

3.2 Physical Network Model

In a wireless network, based on the existing virtualization technology, a physical node can be virtualized as multiple independent virtual wireless nodes, and each virtual wireless node works only in its corresponding virtual wireless network topology. Meanwhile, a small number of mobile nodes scarcely face serious energy problems. Furthermore, multiple channels and multiple radio interfaces technology make it possible to realize the wireless network virtualization. In the wireless network, without considering the limitation of interface and transmission power, we assume that each virtual node corresponds to a virtual interface and each virtual interface has complete physical layer and medium access control layer. Therefore, we can establish link connection by choosing appropriate working channel. Besides the resources that the wired nodes hold, wireless nodes have the channel (spectrum) limitation. For given frequency spectrum resource and each wireless node equipped with multiple wireless interfaces, allocating the same channel of nodes in the same collision domain may lead to the interference on the wireless interface. When a signal is sent or received, the interference could affect the quality of the signal and ultimately result in a decline in the performance of virtual network topology.

The establishment of link in wireless network is based on the allocated wireless channel and the activated node. The capacity of wireless link is related to the quality of wireless channel, including the interference between links and the degree of signal attenuation in the wireless medium. Thus the wireless link capacity is limited and variable, which makes the wireless virtual network embedding problem complex and difficult. In order to lessen the computational complexity, in this paper, we assume that link capacity is fixed. Because the establishment of link is related to wireless channel, the wireless link allocation is constrained by channel and interference.

We model the underlying physical wireless network as an undirected graph $G^S = (N^S, L^S, \mathrm{R}elia(L^S), L^I)$, where N^S is the set of underlying physical network nodes, L^S is the set of the underlying links, $Relia(L^S)$ is the bearing reliability of the link from physical nodes i to physical nodes j (the bearing reliability is evaluated by packet loss rate that is obtained by periodic probe packet, and the lower the packet loss rate is the higher the reliability will be), L^I is the link interference set. The interference among inter-nodes link $l^s \in L^S$ is defined based on the interference model, that is, the set of the interference link is $l^I \in L^I$ within the scope of interference domain. $CPU(n^s)$ is the calculation ability of underlying physical node n^s. We suppose that the number of available channels of the underlying physical network is k, in other words, the available channel set of the physical nodes is $ch(n^s)$, where $ch(n^s) = \{ch_1, ch_2, \cdots, ch_k\}$, and all the channels are orthogonal to each other. Each node possesses infinite wireless interface. The underlying wireless link $l^s \in L^S$ indicates that two wireless nodes have the ability to communicate with each other. Physical network model is as shown in Fig. 1(b). For example, the CPU of physical node A is 60, the bandwidth of the physical link between node A and node B is 30, and the reliability of the physical link between node A and node B is 0.91, whose range is [0,1].

3.3 Interference Model

The unique characteristic of wireless network is the interference arising in the process of parallel transmission of the links. We adopt the matrix $I = |L^S| \times |L^S|$ to refer to the interference among the underlying wireless links. $I_{ij} = 1$ represents the interference between link i and link j. $I_{ij} = 0$ represents that there is no interference between link i and link j. The interference matrix model depends on the physical layer and MAC layer technology. We imitate wireless physical network interference which is based on space distance interference model, and it defines CF as the interference number of an underlying physical network link. The detailed definition is shown in Definition 1.

Definition 1. *For any two nodes n^s and n'^s, $D(n^s, n'^s)$ is the distance between them. For any two links l^s and l'^s, $D(l^s, l'^s)$ is the distance between them, where $D(l^s, l'^s)$ is the shortest distance from any endpoints of l^s to any endpoints of l'^s. When the conflict distance D_{cf} is given, the total conflict number CF after the allocation of virtual network is defined as Eq. (1):*

$$CF = \sum_{e \in C[E]} |\{l^s \in L^S | l^s \neq l'^s \cap D(l^s, l'^s) < D_{cf} \cap (ch_i \oplus l^s = ch_j \oplus l'^s) \cap i = j\}| \tag{1}$$

where e are the links of two-hop path and $C[E]$ is the set of two-hop path links. CH_i and CH_j are the channels, $(i, j = 1, 2, 3 \cdots)$. ch_i is the exclusive-or with l^s, i.e., if link l^s transmits data in channel i, it is 1; otherwise, it is 0. The same applies to ch_j, which is the exclusive-or with l'^s.

When the distance between two different physical links l^s and l'^s is less than conflict distance D_{cf} and the two links are assigned to the same channel $((ch_i \oplus l^s = ch_j \oplus l'^s) \cap i = j)$ at the same time, the total conflict number CF is increased.

3.4 Nodes with Uniform Available-Channel Set

The establishment of link connection between two nodes in underlying wireless network is based on activating common channel. Multiple common channels in wireless network indicate that more channel degrees (see Sect. 3.5) carry more wireless links. The uniform available channel $(availch(n_i))$ of two nodes is defined in Eq. (2):

$$SameCh = \{availch(n_1) \cap availch(n_2) | n_1, n_2 \in N^S\} \tag{2}$$

where $SameCh$ denotes shared channels of physical nodes n_1 and n_2, and the size of $SameCh$ is defined in Eq. (3):

$$NumSameCh(n^s, n'^s) = sizeof(SameCh) \tag{3}$$

where $NumSameCh(n^s, n'^s)$ is the number of shared channels between physical nodes n^s and n'^s. Different channel allocation choices based on channel set of collision domain are shown in Fig. 2. CH_1 and CH_2 are chosen to be allocated to virtual links. Each physical node owns two channel set: an available channel set R and an allocated channel set E. For instance, the available channel set of node A is $R = \{2, 3, \cdots, K\}$and the allocated channel set of node A is $E = \{1\}$, which means CH_1 has been allocated and cannot be allocated.

3.5 Average Channel Degree

The premise of building a wireless link between two nodes is to allocate the same channel to them, that is, the channel degree of physical node n^s is the average of the total number of shared channels that directly connected with the node. Assuming that the physical node n^s is connected with m nodes, its neighbor nodes set is $Nb(n^s)$. Then, the average channel degree of physical node n^s is defined as Eq. (4):

$$ChDegree(n^s) = \sum_{n'^s \in Nb(n^s)} NumSameCh(n^s, n'^s)/m \tag{4}$$

3.6 Link Degree

The connection established between two nodes indicates that the two nodes are in the communication range of each other. Thus, the link degree is the number of

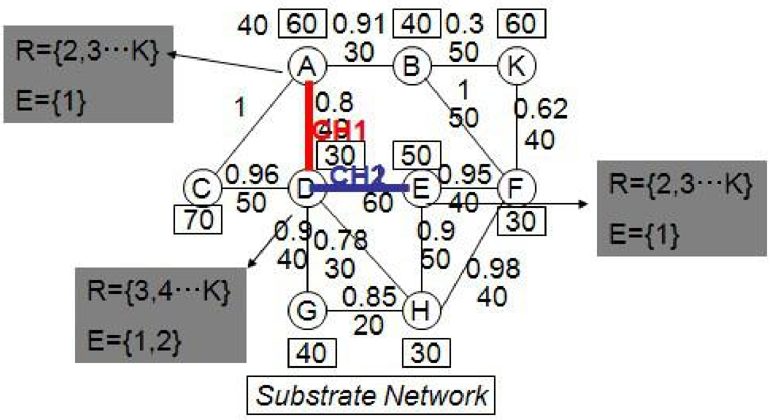


Fig. 2. Different channel allocation choices based on collision domain

wireless links that the node could establish interaction with its neighbor nodes. We suppose that the physical node n^s can communicate with h neighbor nodes, the bearing degrees ($SnLinkDegree$) of the physical node could be represented by the product of link degree h and CPU resource of physical node n^s, which is defined as Eq. (5):

$$SnLinkDegree = h \times CPU(n^s) \tag{5}$$

3.7 Analysis of Interference Effect Weight

The weight of interference effect in underlying physical network is the interference density that the underlying link impacts on other links based on interference model, which is represented by $CfWeightI(l)$. If link l is allocated a bandwidth resource unit, $CfWeightI(l)$ is the interference weight based on interference model, which is resulted from the wireless link. The interference weight of the underlying path is the sum of the interference weight of all physical links, and it is described as Eq. (6):

$$CfWeightI(l) = CF/bandwidth(l) \tag{6}$$

where $bandwidth(l)$ is the bandwidth of physical link l, and CF is defined by Eq. (1).

4 Virtual Network Embedding Algorithm Based on Link Interference (WVNEA-LI)

The interference between wireless links of a wireless network is the main difference from a wired network. The broadcast feature of a wireless link leads to high packet loss rate and severe link interference, which would affect the network throughput. Aiming at the problems that virtual network allocation algorithms in traditional wired networks are not suitable for wireless networks, and the interferences in existing virtual networks after link allocation will react link reliability, in this paper, we propose an optimized algorithm based on the WVNEACLR algorithm [16], which we call it Wireless Virtual Network Embedding Algorithm based on Link Interference (WVNEA-LI). The algorithm not only considers the link reliability and the factor that the link interference may react the communication of wireless link, but also thinks about the low allocation efficiency that is triggered by the separation of nodes and links allocation. A concurrent allocation method is adopted to map nodes and links simultaneously, which effectively coordinates the allocation between the nodes and links.

The WVNEA-LI algorithm includes steps as follows: (1) choose physical network resources according to $VnRequest$; (2) construct interference model based on wireless physical link; (3) generate virtual nodes list, mapping list, allocated virtual nodes list, and allocated channels list; (4) generate physical nodes list and physical neighbor nodes list; (5) allocate nodes and links. The algorithm comprehensively considers link reliability and link interference, applies Dijkstra′s K-shortest path method to find the shortest path, selects different channel in the allocated channel list to each allocated physical link, and employs the concurrent resource allocation methods to effectively coordinate the distribution between the nodes and links. The complete algorithm is shown in **Algorithm 1**.

5 Simulation Results

We use CloudSim2.0 [18] to simulate the basic virtual network embedding algorithm (BVNEA) [3], wireless virtual network embedding algorithm based on link reliability (3 Hops Confined) (WVNEA-LR(3 Hops Confined)) [16] that link allocation limit within 4 jump, wireless virtual network embedding algorithm based on link reliability (Hop-free) (WVNEA-LR(Hop-free)) with no jump constraint allocation. In addition, we evaluate the performance of WVNEA-LI in aspects of acceptance ratio of virtual network, benefit-cost (B/C) ratio, and resource utilization. Table 1 lists the above four algorithms and their abbreviations.

5.1 Simulation Environment

The simulation uses the topological generator BRITE to generate the topology that contains 100 nodes, which is used to simulate a physical network. There is a 50% chance to randomly generate 400 physical links. The packets loss rate of each physical link is different, and it is modified in real time. The computing ability of each physical node is randomly generated with the value range of 10-80 units. Similarly, the bandwidth of each physical link is randomly generated with the value range of 10-70 units. This paper sets the arrival rate of virtual network *VnRequest* as 4 per second, and the life cycle of each *VnRequest* is 10 seconds. Each *VnRequest* is randomly generated with 3 to 10 virtual nodes. Connection probability of each pair of nodes is also 50%. Requirement capability of each virtual node is randomly generated as well with range of 1-30 units, and bandwidth requirement of each virtual link is randomly generated as well with range of 1-20 units.

Algorithm 1. WVNEA-LI Algorithm

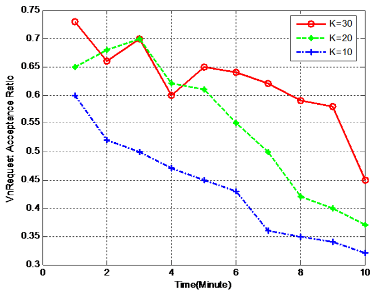
Input: $\{VnRequest_0, VnRequest_1, \cdots, VnRequest_m\}$, $SubNet$.
Output: $mappedVnNodeList$, $cfChannelLinkList$.

```
    /*Initialize*/
    used channel set $cfChannelLinkList = null$
    allocated virtual nodes list $mappedVnNodeList = null$,
    mapping relation list $nodeMappingRelation = null$.
    /*Steps*/
 1: for ($i = 0$; $i \le m$; $i++$) do
 2:     if $cpu(n^s)$ and $bandwidth(l^s)$ satisfy $cpu(n^v)$ and $bandwidth(l^v)$ of $VnRequest_i$
    then
 3:         Map $VnRequest_i$ to $SubNet$;
 4:         break;
 5:     end if
 6: end for
 7: initialize $cfChannelLinkList$, according to $I = |L^S| \times |L^S|$;
 8: sort $n_v$ of $VnRequest_i$ by $cpu(n_v)$ in descending order,
    and put them in $vnNodeList$ successively, $vnNodeList = \{{n^v}_0, {n^v}_1, \cdots, {n^v}_h\}$;
 9: sort $n_s$ of $SubNet$ in descending order based on the product of $ChDgree(n_s)$
    and $SnLinkDegree$, and put them in $snNodeList$ successively, $snNodeList =$
    $\{{n^s}_0, {n^s}_1, \cdots, {n^s}_g\}$;
10: for ($j = 0$; $j \le h$; $j++$) do
11:     for ($k = 0$; $k \le g$; $k++$) do
12:         if $cpu({n^s}_k)$ and $bandwidth({l^s}_k)$ satisfy $cpu({n^v}_j)$ and $bandwidth({l^v}_j)$ then
13:             map ${n^v}_j$ to ${n^s}_k$;
14:             $nodeMappingRelation = nodeMappingRelation \cup \{{n^v}_j \to {n^s}_k\}$;
15:             $mappedVnNodeList = mappedVnNodeList \cup \{{n^v}_j\}$;
16:         else ${n^v}_j$ is connected with the nodes of $mappedVnNodeList$
17:             establish the shortest link based on $relia({l^s}_k)$ and different channel al-
                location within the $cfChannelLinkList$ collision domain;
18:         end if
19:     end for
20: end for
21: return
```

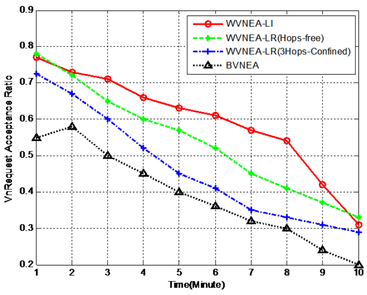
5.2 Results and Analysis

We compare the acceptance ratio of virtual network (virtual network acceptance ratio), B/C, the average utilization ratio of nodes and links as the performance metrics. We evaluate the performance of the four algorithms.

In the simulation, the number of the request nodes is 3 or 4. We compare the acceptance ratio of the virtual network request in the three cases of the available channel $K = 10, 20, 30$ over a period of time. Fig. 3(a) shows that, with the growth of time, *VnRequest* acceptance ratio of WVNEA-LI is increasing with the rising of channel number. However, when time grows the acceptance ratio of the virtual network decreases , the reason is that it is restricted by the channel while allocating links. When there is no common channel between nodes, the establishment of link may fail, and the acceptance ratio of the virtual network requests may be reduced accordingly. Fig. 3(b) shows that the virtual network request acceptance ratio of WVNEACLI algorithms is obviously higher than others. WVNEA-LI algorithm adopts the interference technology on the basis of selecting reliable links. Thus, link allocation will not affect allocated links. The proposed algorithm also makes the physical network stable without affecting the allocation of virtual network.



(a)



(b)

Fig. 3. Virtual network acceptance ratio

As shown in Fig. 4, the benefit-cost ratios of WVNEA-LI, WVNEA-LR(Hop-free) and WVNEA-LR(3 Hops Confined) are similar and considerably better than that of BVNEA because the three algorithms take into account the reliability of the underlying physical network link while constructing virtual networks. The resources allocation is in a relatively concentrated area. Because WVNEA-LI has better virtual network acceptance ratio, the benefit-cost ratio is higher at some time.

Fig. 5 shows that the WVNEA-LR(Hop-free) algorithm and the WVNEA-LR(3 Hops Confined) algorithm have preferable node (Fig. 5(a)) and link (Fig. 5(b)) utilization rate compared with the basic allocation algorithm BVNEA, and the WVNEA-LI algorithm is superior to other three algorithms. The three

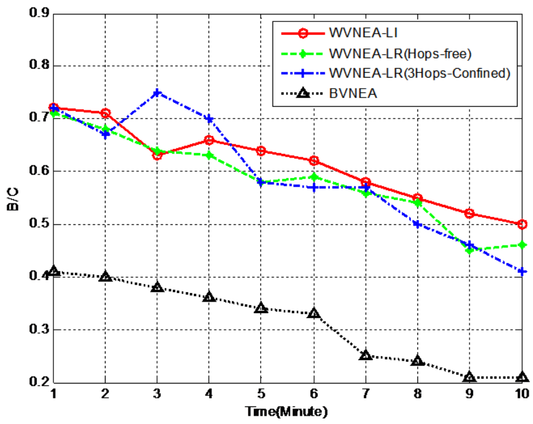


Fig. 4. Benefit cost ratio

algorithms all allow mapping multiple virtual node of the same *VnRequest* to the same physical node at the same time, which inevitably leads to the increase of success rate of the node construction and node utilization rate. In addition, the utilization rate of virtual networks of the WVNEA-LI algorithm is higher than other three algorithms owing to taking link interference into account, so the utilization rate is the most favorable. However, to some extent, it also reduces the link utilization rate and saves link resources, hence the link utilization rate of BVNEA, which barely uses the node re-mapping technology, is closer to the other three algorithms at quite a few time points.

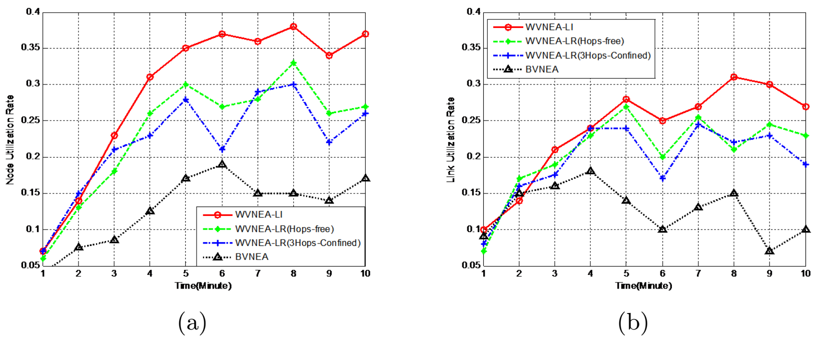


Fig. 5. Average utilization rate

6 Simulation Results

In this paper, we propose a wireless virtual network embedding algorithm based on link interference (WVNEA-LI) to cope with link interference and two-step mapping problem in WDC. The WVNEA-LI algorithm strengthens coordination between nodes allocation and links allocation, and improves the efficiency of allocation in wireless networks by using concurrent allocation and interference matrix. The simulation results show that, by fully considering interference of wireless links, the WVNEA-LI algorithm optimizes the link reliability of virtual network allocation algorithm WVNEA-LR, improves the acceptance ratio of virtual network, increases the network benefit-cost ratio and the resource utilization. However, the deficiency is that the optimization algorithm hardly takes the resource load balancing into account.

In addition, green computing has become one of the most concerning points in information technology industry. The cost of energy consumption occupies a large part of wireless data center budget. Therefore, how to apply wireless virtual resource allocation algorithms to implement energy conservation in WDC is our next research work.

Acknowledgments. This work is partially supported by Program for New Century Excellent Talents in University (NCET-12-0164); National Natural Science Foundation of China (61370094); Natural Science Foundation of Hunan (13JJ1014); National Key Technology R&D Program(2012BAD35B06).

References

1. Halperin, D., Kandula, S., Padhye, J., Bahl, P., Wetherall, D.: Augmenting data center networks with multi-gigabit wireless links. ACM SIGCOMM Computer Communication Review 41, 38–49 (2011)
2. Yun, D., Ok, J., Shin, B., Park, S., Yi, Y.: Embedding of virtual network requests over static wireless multihop networks. Computer Networks 57(5), 1139–1152 (2013)
3. Kolliopoulos, S.G., Stein, C.: Improved approximation algorithms for unsplittable flow problems. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, pp. 426–436. IEEE (1997)
4. Zhu, Y., Ammar, M.H.: Algorithms for assigning substrate network resources to virtual network components. In: INFOCOM, pp. 1–12 (2006)
5. Yu, M., Yi, Y., Rexford, J., Chiang, M.: Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Computer Communication Review 38(2), 17–29 (2008)
6. Smith, G., Chaturvedi, A., Mishra, A., Banerjee, S.: Wireless virtualization on commodity 802.11 hardware. In: Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, pp. 75–82. ACM (2007)
7. Shrestha, S.L., Lee, J., Chong, S.: Virtualization and slicing of wireless mesh network. In: International Conference on Future Internet Technologies (2008)
8. Fan, J., Ammar, M.H.: Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In: INFOCOM (2006)

9. Cui, Y., Wang, H., Cheng, X.: Wireless link scheduling for data center networks. In: Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, pp. 4. ACM (2011)
10. Cui, Y., Wang, H., Cheng, X., Li, D., Yla-Jaaski, A.: Dynamic scheduling for wireless data center networks. IEEE Transactions on Parallel and Distributed Systems 24(12), 2365–2374 (2013)
11. Cui, Y., Wang, H., Cheng, X., Chen, B.: Wireless data center networking. IEEE Wireless Communications 18(6), 46–53 (2011)
12. Shin, J.Y., Sirer, E.G., Weatherspoon, H., Kirovski, D.: On the feasibility of completely wireless datacenters. In: Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 3–14. ACM (2012)
13. Szeto, W., Iraqi, Y., Boutaba, R.: A multi-commodity flow based approach to virtual network resource allocation. In: Global Telecommunications Conference, GLOBECOM 2003, vol. 6, pp. 3004–3008. IEEE (2003)
14. Ahn, S.W., Yoo, C.: Network interface virtualization in wireless communication for multi-streaming service. In: 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE), pp. 67–70. IEEE (2011)
15. Paul, S., Seshan, S.: Technical document on wireless virtualization. GENI: Global Environment for Network Innovations, Tech. Rep. (2006)
16. Luo, J., Liu, C., Li, R.: Wireless virtual network allocation based on link reliability. Journal on Communications 33, 88–95 (2012)
17. Lv, P., Wang, X., Xu, M.: Virtual access network embedding in wireless mesh networks. Ad Hoc Networks 10 (2012)
18. Buyya, R., Calheiros, R.N., Beloglazov, A.: Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services [ol], `http://www.cloudbus.org/cloudsim`

A Weighted Centroid Based Tracking System in Wireless Sensor Networks[*]

Hongyang Liu[1], Qianqian Ren[1,2,**], Longjiang Guo[1,2], Jinbao Li[1,2], Hui Xu[1] , Hu Jin[1], Nan Wang[1] and Chengjie Song[1]

[1] College of Computer Science and Technology, Heilongjiang University, Harbin, China
[2] Key Laboratory of Database and Parallel Computing of Heilongjiang Province, Harbin, China
{swisa.liu,rqqhlju}@gmail.com

Abstract. This paper designs a target tracking system in wireless sensor networks. Considering the special requirements of target tracking, we adopt a sleeping schedule into the tracking procedure to reduce the number of nodes. To describe the trajectory of the target, we propose a tracking model based on random walk of Markov Chain. To simplify the complexity of localization, we incorporate two efficient weighted methods into our centroid localization algorithm. The experimental results show the efficiency of our system.

Keywords: Target tracking, random walk, centroid, network lifetime.

1 Introduction

Wireless sensor networks (*WSN*s) have completely affected and changed the way we live and work. In the network, a large number of tiny and inexpensive sensor nodes are deployed and used to monitor highway traffic, wildlife habitat conditions, forest fires, military battlefield and so on[1,2]. In many applications of sensor networks, target localization is a fundamental function, especially for the applications such as monitoring the movement of enemy vehicles on the battlefield, monitoring forest fires and the spread, detecting states of the vehicles on the highway[3.4]. However, there are characteristics of *WSN*s has proposed challenges for it. In sensors networks, the lifetime of nodes is a hard problem, as the sensor nodes are battery powered. To reduce the energy consumption of sensor nodes, many work introduce a series of

[*] This paper has been supported by the National Natural Science Foundation of China under Grant No. 61300225 and 61370222; the Science and Technology Research of Heilongjiang Education Office under Grant No.12521391; the Program for New Century Excellent Talents in University under Grant No.NCET-11-0955; the Programs Foundation of Heilongjiang Educational Committee for New Century Excellent Talents in University under Grant No.1252-NCET-011; the Program for Group of Science and Technology Innovation of Heilongjiang Educational Committee under Grant No.2013TD012 and the Natural Science Foundation of Heilongjiang Province under Grant No. F201225.
[**] Corresponding author.

energy efficient mechanisms[5,6,7]. In these methods, making nodes work in turn is a very efficient way. At a given moment, only a small number of nodes to work and others sleep. The nodes that detect the target (that's the target is in their sensing range) work. Initially, the sensor nodes, which situated around the door of the laboratory, are working and others are sleeping. When the target was on the sensing range of the working sensor nodes, these nodes woke up their neighbor nodes. The working nodes enter into the sleep state automatically when the target is out of the sensing range of the working nodes. It is the most saving energy scheme, if set to all nodes in sleeping. However, this sleeping based mechanism cannot achieve real-time monitoring, locating and tracking. As nodes in sleeping state can't detect the target when the target enters into their sensing range. Furthermore, these sleeping nodes cannot receive the information of the target as well. Considering the requirements of energy saving in target tracking, we design an energy efficient tracking system in wireless sensor networks. Our system adopts a series of weighted centroid methods, which are adaptive to different situations. To reduce the energy cost of nodes, we design a sleeping scheduling mechanism, which makes part necessary nodes work and others sleep.

The main contributions of this paper are as follows: 1) This paper introduces a target moving model, which adopts the idea of random walk of Markov Chain. In order to simulate the real trace more accurately, we further choose the moving direction of the target in a random way. 2) We design a sleeping schedule to save energy consumption of nodes, which increases the lifetime of the network efficiently. 3) We propose a mechanism to record the pattern of the target movement in order to predict the target trajectory. 4) We incorporate existing weighted centroid methods into our system. 5) We simulate a large laboratory scale environment, and verify the excellent performance of our system.

Distribution of the remaining part of this paper is as follows. Section 2 reviews the related work. Section 3 describes the system model. Section 4 presents the detail of our system design. In Section 5, we provide experiments to analyze and verify the performance of the proposed algorithm. We conclude our paper in Section 6.

2 Related Work

Target tracking in different applications of sensor networks, plays a very prominent role in areas such as environmental surveillance, intelligent transport, disaster predication and location-based services[8,9,10,11]. However, the range and functionality of the sensors are quite different in different applications. It has been proposed many different kinds of tracking methods and data model processing methods[12,13,14,15]. For example, the authors in [14] give a minimum contour-tracking algorithm to minimize the number of working nodes with guarantee of tracking quality[15].

There are two major problems in wireless sensor networks, energy consumption and tracking accuracy. Many excellent ideas have been proposed to track the target with a guarantee of energy conservation and accuracy[16,17,18,19,20,21]. Such as the method of binary sensor tracking in sensor networks, robustness and efficiency of this model in the large-scale sensor networks is good[22]. This provides a great deal of confidence for we use binary sensors. At the meanwhile, the binary sensor saves energy because the binary sensor only provides one bit information that the target presence or absence at their range of sensing[23]. However, the paper does not consider the energy consumption issue that the sensors are not involved in a location in every moment. This paper will propose a work and sleep state mode to save energy consumption issues above.

Most of the existing works are based on the assumptions that the movement of the target is regular linear or curvilinear with a constant speed. In our paper, we adopt the idea of random walk of Markov Chain to describe the trajectory of the target, in order to make it more close to the target trajectory in real life. As the random property of random walk, the target movement can be regular or swing line. We also set a "face" for the target to ensure that it always moves toward the forward direction, and its movement is not random totally. We will describe it in details in later sections.

3 System Model

3.1 Network Model

It is assumed that the network is consisted of a large number of binary sensor nodes. Those binary sensor nodes provide one bit information that the target presence or absence at their sensing range[23]. We further suppose that the output of each node has two values: When the target is in its sensing range, the node outputs 1; otherwise, the node outputs 0.We assume that the sensing radius of all nodes is same, and the radio radius of each node is same as its sensing radius.

3.2 Target Trajectory Model

In this paper, we use the idea of random walk of Markov Chain to describe the trajectory of the target. First of all, we review the definition and several significant and useful properties of Markov Chain as literature [24].

Let$\{Y_n,n=0,1,2,...\}$be a stochastic process. If $Y_n=i$, the process donates it is in state i at time n. We suppose the P_{ij} is the probability from state i to state j. That is,

$$P\{Y_{n+1}=j|Y_n=i,Y_{n-1}=i_{n-1},\ldots,Y_1=i_1,Y_0=i_0\}=P_{ij} \tag{3.1}$$

Equation (3.1) is the definition of Markov chain. We can get the state Y_{n-2} is independent of the stateY_{n-1} and the past state Y_{n-1} is independent of the present state Y_n. Meanwhile, P_{ij} is a probability and it is nonnegative so that there must be a transition into other states, so we have,

$$P_{ij}\geq 0,i,j\geq 0;\sum_{j=0}^{\infty} P_{ij}=1,\text{i}=0,1\ldots \tag{3.2}$$

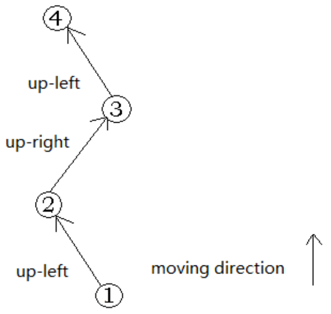
We can get two major properties in equation (3.1) and equation (3.2), the independence of Markov Chain and the probabilities is nonnegative. Next, we obtained a random walk model naturally.

Definition 1(Random Walk Model). A Markov chain is said to be a random walk if the states are i=0,±1,±2,... for $0<p<1$,

$$P_{i,i+1}=p=1-P_{i,i-1}, i=0,\pm1,\dots \quad (3.3)$$

We may think of the case of (3.3) as a drunk man, who is walking straight with probability p to turn left and with probability $1-p$ to turn right.

There are 2^n different kinds of conditions when the target takes the random walk. For n is the timestamp of the target trajectory. Figure 1 and figure 2 depicts an example of a random walk respectively. The circle indicates the position of the target at a timestamp, and the number in the circle indicates the continuous time sequence. In figure 1, we can see that the target movement direction from timestamp 1 to the timestamp 2 is up-left. As the target has the same probability to turn left or right in every step. So that, the other condition from timestamp 1 to the timestamp 2 is up-right in figure 2. It is the same to that from timestamp 2 to the timestamp 3 and later consequence. According to the direction of every step, we can summarize a moving pattern of the target. We will describe it in section 4 in details.



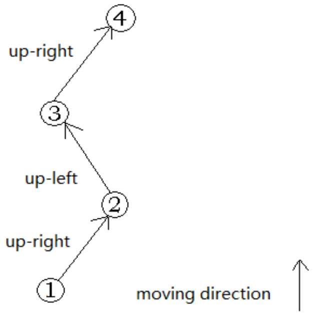


Fig.1. An example of the random walk **Fig.2.** An example of the random walk

4 A Weighted Centroid Tracking System

In this section, we will give the detail of the design of our tracking system. This system consists of the following five modules: moving target discovering module, target localization module, target information transmission module, target trajectory describing module and target movement prediction module. We will give the details of each module in the following sections.

4.1 Target Discovering Module

When the target as entered into the sensing region around the door, those sensor nodes with a certain sensing radius around the target will discover the target, collect the sensed data(such as accelerated attribute) and change the output values of the nodes. To reduce the energy cost of nodes, we set the sensor nodes around the door work while other nodes sleep initially.

Figure 3 depicts the procedure of our system. We use the laboratory as a background. In the figure, nodes are distributed in the sensing region uniformly. The red nodes are working nodes.

When the target enters into the sensing region, the red nodes can discover the target and wake up their neighbor nodes. For example, in the figure, the node 81 has five neighbor nodes, that are node 71, 72, 82, 91, and node 92. Node 81 broadcasts waking message to its neighbors. When the neighbor nodes receive the waking message, they begin to work and collect the sensed data of the target. As the target moving, the nodes surrounding the target are waken and turn to work state. As shown in Figure 4,

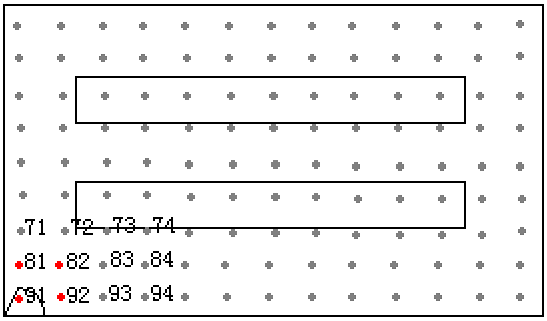


Fig. 3. The figure of discovering target model without target

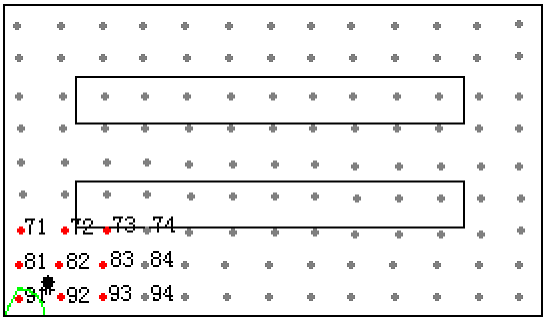


Fig. 4. The figure of discovering target model with target

When the target moving out of the sensing range of the working nodes, those red nodes changes to the sleep state, which turns to grey. For example, in the figure5, the target moving out of the sensing range of node 91, 92 and 93. Then their color changes to grey automatically, which indicates they are turning to sleeping state from working state.

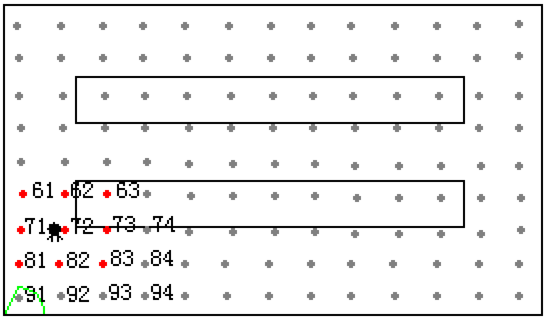


Fig. 5. The figure of discovering target model as target moving

4.2 Target Localization Module

When receive the sensed data, the sensors can locate the target using certain localization algorithms. Many existing algorithms can be adaptive to our system. In this paper, we use the idea of centroid localization methods in literature[23], which gives two weighted methods, that are Pessimistic algorithm and Distance to the Path algorithm respectively. These two methods present two different weight computation methods. Because we use a thought of random walk Markov chain, which gives a guarantee of the target in a short term (at least within the one-step), the target has a straight-line trajectory. Thus, we can use the successive refinement to obtain a more accurate tracking trajectory[23]. We will describe the two localization algorithms as followings.

4.2.1 Pessimistic Algorithm

Assumed that the sensing radius of the sensors are R, so we use $\frac{1}{R}$ as weights. Unlike weighted uniformly method is the gradual refinement steps: in figure 6, assume that, the entry point of the target is t_0, the out point is t_i, and the current point is e. Let d_i presents the length of secant $\overline{t_o t_i}$, $\cos\alpha = \frac{d_i}{2R}$, for $0<e<i$, $r_e^2=R^2+d_e^2-2Rd_e \cos\alpha$ $=R^2+d_e^2 d_e d_i$. The new weight at e is $\frac{1}{r_e}$.

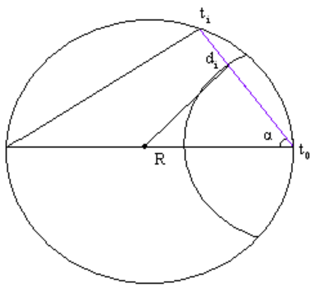


Fig. 6. Pessimistic algorithm refinement schematic diagram

```
Algorithm 1: Pessimistic Algorithm
1: Record the enter point $t_0$ and estimate the current point $e_1$ by the weights of $\frac{1}{R}$.
2: Record the out point $t_i$. Calculate $d_i$, that is the length of $\overline{t_o t_i}$.
3: Use $\frac{1}{r_e}$, that is the reciprocal of $r_e^2=R^2+d_e^2-2Rd_e\cos\alpha = R^2+d_e^2 d_e d_i$ to
      successive refine the trajectory.
```

4.2.2 Distance to the Path Algorithm

The same as Pessimistic Algorithm, we record the entry point t_0, the out point t_i, and the current point e. Let d_i presents the length of secant $\overline{t_o t_i}$, but our new weight is about the distance from the center of the sensor to d_i, which is the point half of d_i exactly, as shown in Fig.7. Let h denotes the distance from R to d_i. We calculate the path distance $h=\sqrt{R^2-(\frac{d_i}{2})^2}$ and use $\frac{1}{h}$ as weights. For any point on the path, the weight is $\frac{1}{h}$.

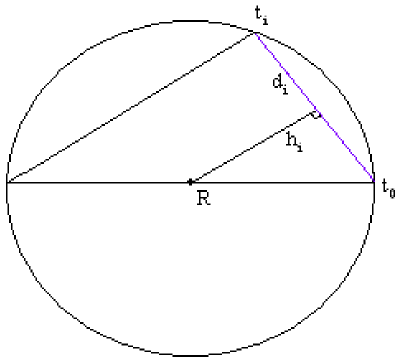


Fig. 7. Distance to the Path Algorithm schematic diagram

```
Algorithm2: Pessimistic Algorithm
1: Record the enter point$t_0$, estimate the current point $e_1$ by the weights of $\frac{1}{R}$.
2: Record the out point $t_i$. Calculate $d_i$, that is the length of $\overline{t_o t_i}$.
3: Calculate the path distance $h = \sqrt{R^2 - (\frac{d_i}{2})^2}$.
4: Use $\frac{1}{h}$ as new weights.
```

4.3 Target Information Transmission Module

When nodes collect the sensed data of the target, they need to transmit data to Sink for further processing. Many researches adopted the directed diffusion[25,26], LEACH [27], and two-tier data dissemination (TTDD) [26]. In this paper, we adopt an algorithm based on routing tree, which is the simplest routing structure and meanwhile the routing tree structure. Moreover, it can match to the localization algorithms well. As the target moving, we pruned these nodes, which are in the tree and far away from the target. Multiple nodes surrounding the target collaborate to make the collected information more complete, reliable, and accurate[25].

4.4 Target Movement Prediction Module

Many works have been done on target movement prediction. A naive idea is that the current position can be formed on the last point and the current point. So that we can construct a line, and this line is the walking pattern of the target. However, this idea is based on the assumption that the moving direction of the target is constant. It is not realistic in the real applications. In this system, we adopt the property of the random walk of Markov Chain, where the next movement of the target is independent with the last point. Therefore, we can simplify this procedure as recording two moving direction information of the target. We just need to record the position of the target at the current timestamp, one timestamp before and two timestamp before. Based on these information, we can conclude the moving direction of the target next timestamp.

Figure 4 illustrated the procedure of our algorithm. The circle indicates the position of the target at a timestamp, and the number in the circle indicates the continuous time sequence. We can see that the target movement direction from timestamp 1 to the timestamp 2 is up-left. Therefore, that the upper left is the first recording. From timestamp 2 to timestamp 3, the target chooses up-right direction. Our second record is up-right. We use the accumulator to record the information. According to the above method, we can summarize a moving pattern of the target. That is moving toward the up-left first, and then moving toward up-right. We count the times of the pattern appearance.

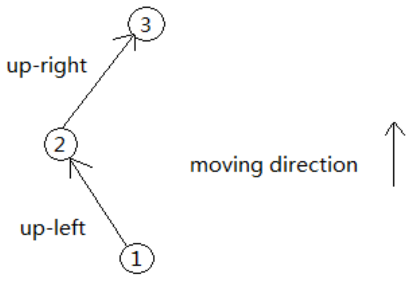


Fig. 8. Records the movement of the moving target schematic diagram

Next, we write the times of all patterns appearance into the target movement prediction file. As the target moving, we can get the current direction easily. Then we will predict the direction of the target trajectory by traversal in the target movement prediction file and find the max times of the next direction from the current direction. The value of the times means the probability that the target moves as the corresponding direction. Therefore, the maximum time is the most likely direction in which the target will choose next time. We will obtain more accurate estimate since calculated using the two iterations.

4.5 Target Trajectory Describing Module

In this module, we will show the trajectory generated by the random walk on screen. We had saved the coding of the track to the document in advance. While calling the display module, the system reading the position coordinates from a file, and then marked the coordinates of each target in the figure. Link these coordinates to draw a path in this form, which will show in the background.

5 Experiments and Evaluations

To validate the algorithm we designed, we simulated experiments on xp and win7 system using vc++6.0. We set the experiments on a 1000×500units laboratory environment, where 72 nodes are distributed uniformly. The distance between each node is 100units, and the sensing radius of each node is 150 units. The speed of the target movement is 50 unit/s. A plan view of the laboratory is illustrated in Figure 9. We can see that the color of sensor nodes around the door is red, which means that they are in a working state, while other sensor nodes are gray means they are in the sleep state. We set a "face" for the target to make sure that it always moves toward the forward direction, and its movement is not random.

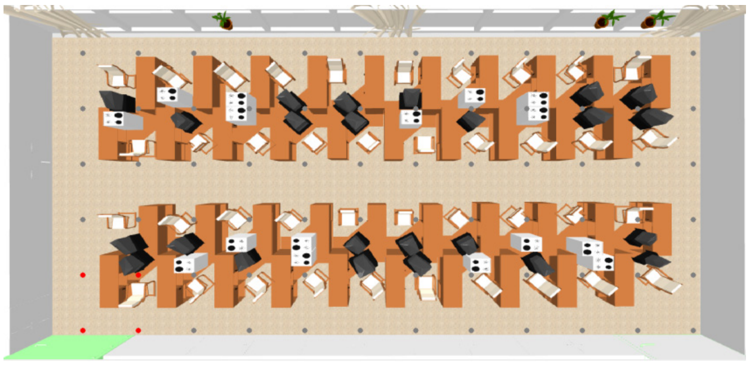


Fig. 9. Plan view of simulation laboratory with node distribution

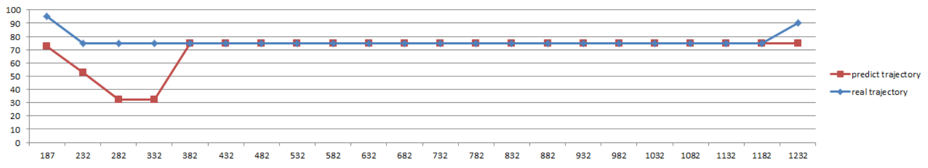
We simulate two groups of experiment with constant speed of the target movement are 50 unit/s and 30 unit/s. Each group verified straight line and folds linear movement of the target respectively. The fold linear movement divided into swing movement situation and a relatively regular movement situation. The results of the trajectory prediction experiment are shown in figure 10. In the figure, the horizontal axis represents the value of *x*-coordinate and the vertical axis represents the value of *y*-coordinate. We set the start point of the predicted trajectory is (0,0).

During a path with a constant direction of continuous movement, the accuracy of the prediction is very well, almost exactly, no matter how fast the speed is. As shown in a) and b).

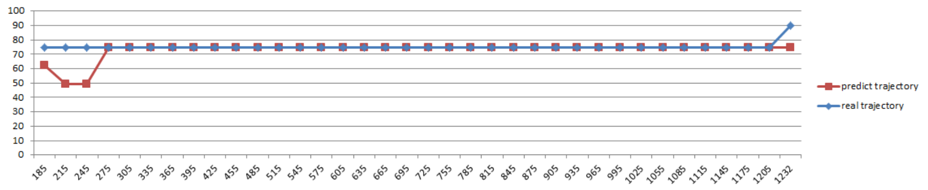
We can see that there is only a small deviation in c) and d). That is easy to understand due to the random walk of Markov Chain characteristics and the pattern we proposed above. We recorded two successive changes of direction (state transition) case. The target changes the direction constantly, which is erratic and there is no pattern to follow no matter how many points and how fast the speed of the target we set. This will inevitably lead to the deviations of predict. However, deviation in our method is relatively small and it is within the acceptable error range.

At the same time, the real regular fold linear trajectory compared with the predict trajectory shows better performance as shown in e) and f). The movement of the target follows the pattern, is the best practice for we record the pattern of the target trajectory, which shows our method is great, again.

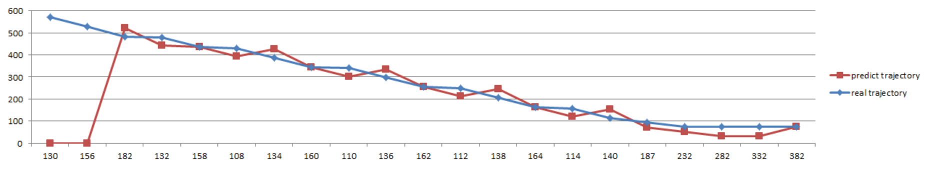
The predict trajectory is close to the real tractor in the figure 10. In some cases, the accuracy is not very high, but they are within the error range. Therefore, when the movement of the target is constant direction line or relatively regular fold line, the accuracy of our proposed algorithm is high. Furthermore, for the swing fold linear movement of the target, lower speed will lead to a better result. Overall, our algorithm showed a better performance.



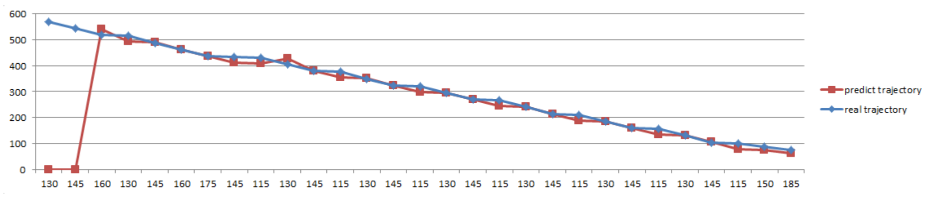
a)The real straight trajectory compared with the predict trajectory, speed is 50uint/s



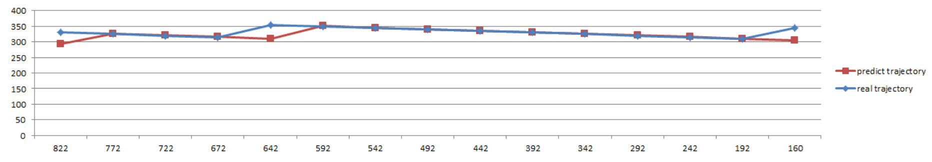
b)The real straight trajectory compared with the predict trajectory, speed is 30uint/s



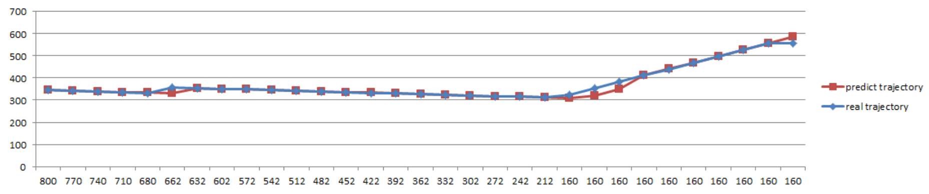
c)The real swing curvilinear trajectory compared with the predict trajectory, speed is 50uint/s



d)The real swing curvilinear trajectory compared with the predict trajectory, speed is 30uint/s



e)The real regular curvilinear trajectory compared with the predict trajectory, speed is 50uint/s



f)The real regular curvilinear trajectory compared with the predict trajectory, speed is 30uint/s

Fig. 10. Real trajectory compared with the predict trajectory

6 Conclusions and Future Work

This paper adopts the simplicity of binary sensors and designs a tracking system. It first uses the randomness of Markov chain's random walk to describe the target trajectory. Second, we incorporate two weighted centroid localization algorithms into our target localization procedure that are Pessimistic and Distance to the Path algorithms respectively. Third, we take the life problem of the node into account, and use working and sleeping state schedule to save energy cost. Our target trajectory model is flexible and has more randomness than regular linear movement. Moreover, the trajectory prediction performs well. Experiments show that our system is stable, reliable, and has high robustness.

References

1. Li, Z., Nadon, S., Cihlar, J.: Satellite detection of Canadian boreal forest fires: development and application of the algorithm. International Journal of Remote Sensing 21(16), 3057–3069 (2000)
2. Duarte, M.F., Hu, Y.H.: Vehicle classification in distributed sensor network. Parallel Distributed Computing 64(7), 826–838 (2004)
3. Donovan, B., McLaughlin, D.J., Kurose, J.: Principles and design considerations for short-range energy balanced radar networks. In: IGARSS 2005 (2005)
4. Guo, G., Zhou, M.: Using MODIS Land Surface Temperature to Evaluate Forest Fire Risk of Northeast China. IEEE Geoscience and Remote Sensing Letters 1(2) (April 2004)
5. Ramanathan, R., Rosales-Hain, R.: Topology Control of Multihop Wireless Networks using Transmit Power Adjustment. In: Proceedings of IEEE Infocom 2000. Tel Aviv (March 2000)
6. Chang, J.-H., Tassiulas, L.: Energy Conserving Routing in Wireless Ad-hoc Networks. In: Proceedings of IEEE Infocom 2000. Tel Aviv, Israel (March 2000)
7. Heinzelman, W.R., Kulik, J., Balikrishnan, H.: Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In: Proceedings of MobiCom 1999, Seattle, WA (August 1999)
8. He, T., et al.: Achieving Real-Time Target Tracking Using Wireless Sensor Networks. ACM Trans. on Embedded Computing System (2007)
9. Li, D., Wong, K., Hu, Y.H., Sayeed, A.: Detection, classification and tracking of targets in distributed sensor networks. IEEE Signal Processing Magazine (March 2002)
10. Zhao, F., Shin, J., Reich, J.: Information-driven dynamic sensor collaboration for tracking applications. IEEE Signal Processing Magazine (2002)
11. Brooks, R.R., Ramanathan, P., Sayeed, A.M.: Distributed target classification and tracking in sensor networks. Proc. of the IEEE 91, 1163–1171 (2003)
12. Cheng, X., Thaeler, A., Xue, G., Chen, D.: TPS: A time-based positioning scheme for outdoor wireless sensor networks. In: INFOCOM (2004)
13. Ding, M., Chen, D., Xing, K., Cheng, X.: Localized fault-tolerant event boundary detection in sensor networks. In: INFOCOM (2005)
14. Jeong, J., Hwang, T., He, T., Du, H.C.: Mcta: Target tracking algorithm based on minimal contour in wireless sensor networks. In: INFOCOM, pp. 2371–2375 (2007)

15. Ren, Q., Li, J., Cheng, S.: Target Tracking under Uncertainty in Wireless Sensor Networks. In: 2011 Eighth IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (2011)
16. He, T., et al.: An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In: Mobisys (2004)
17. Wang, Z., Bulut, E., Szymanski, B.K.: Distributed energy efficient target tracking with binary sensor networks. ACM Trans. Sen. Netw. 6, 32:1–32:32 (2010)
18. Zhong, Z., Zhu, T., Wang, D., He, T.: Tracking with unreliable node sequences. In: INFOCOM (2009)
19. Ding, M., Cheng, X.: Fault tolerant target tracking in sensor networks. In: MobiHoc 2009: Proceedings of the Tenth ACM International Symposium on Mobile ad Hoc Network in Gand Computing, pp. 125–134. ACM, New York (2009)
20. Ren, Q., Li, J., Gao, H.: Tpss: A two-phase sleep scheduling protocol for object tracking in sensor networks. In: MASS, pp. 458–465 (2009)
21. Boukerche, A., Cheng, X., Linus, J.: Energy-aware data-centric routing in microsensor networks. In: Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (2003)
22. Boukerche, A., Cheng, X., Linus, J.: Energy-aware data-centric routing in microsensor networks. In: Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (2003)
23. Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., Miyashita, M.: A line in the sand: A wireless sensor network for target detection, classification, and tracking. Comput. Networks 46(5), 605–634 (2004)
24. Kim, W., Mechitov, K., Choi, J.-Y., Ham, S.: On target tracking with binary proximity sensors. In: Proc. IPSN (2005)
25. Ross, S.M.: Introduction to Probability Models. Academic Press (2011)
26. Zhang, W., Cao, G.: IEEEDCTC: Dynamic Convoy Tree-Based Collaborationfor Target Tracking in Sensor Networks. In: IEEE (2004)
27. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication. In: Proc. Mobile Computing and Networking, pp. 56–67 (August 2000)
28. Krishnamachari, B., Estrin, D., Wicker, S.: Modeling data-centric routing in wireless sensor networks. USC Computer Engineering Tech. Rep. CENG 02–14 (2002)

A Smartphone Location Independent Activity Recognition Method Based on the Angle Feature

Changhai Wang, Jianzhong Zhang, Meng Li, Yuan Yuan, and Yuwei Xu

College of Computer and Control Engineering
Nankai University, Tianjin, China
{storm_xp2008,limeng,yuany}@mail.nankai.edu.cn,
{zhangjz,xuyw}@nankai.edu.cn

Abstract. The smartphone-based human activity recognition method is helpful in the context awareness, health monitoring and inertial positioning. Comparing with the traditional wearable computing which fixes accelerometers on the specific positions of a user body, the activity recognition method based on a smartphone faces the problem of varying sensor locations. In this paper, we lay emphasis on the study of a feature extraction algorithm which is independent of the phone locations. First, the angle motion model is presented to illustrate the human activities. The model describes the difference among walking, going upstairs and going downstairs. Then, an angle feature extraction algorithm is proposed according to the angle motion model. Our analysis shows that different activities have significantly different angle features. Finally, our experiments are introduced. The experiments include data collecting, analysis of experiments results. The experiments results show that the recognition accuracy improved by 2% through adding the angle feature to original features.

Keywords: Smartphone, accelerometer, activity recognition, location independent, angle feature.

1 Introduction

There were ever-increasing researches in the field of human activity recognition in the last 20 years. In these researches, motion data are collected through accelerometers which are fixed on human's body, and these data are used to predict user's activity. These researches are widely used in health monitoring, elderly care and energy expenditure estimation. With the development of micro-electrical mechanical systems, accelerometers are miniaturized so that they can be embedded into mobile devices.

In addition to these traditional wearable computing functions, smartphone-based activity recognition also gives opportunities for some emerging applications. In recent years, with the increase of smartphones and PADs, Mobile Internet is coming. In the course of the development of Mobile Internet, it is important to mine user's habits and perceive surroundings information at any time. This kind of information is essential for developing excellent Mobile Internet products and enhancing application's user stickiness. As the most important personal information, user's activity has a pivotal position in digging user's habits and perceiving environment information. For example, we can speculate user's routines, career and other personal information through records of daily activities. We can guess current location through analyzing user's activity. All the information related to the user and the environment can be used on personalized recommendation, advertising target delivery and so on.

However, smartphone based activity recognition has its own characteristics. Compared with traditional wearable computing, activity recognition based on smartphone faces a new problem. In traditional wearable computing, the accelerometers are often fixed at specific body positions. In the smartphone based activity recognition, the user may put his phone in different pockets, or even in the hand. Research shows that, the collected acceleration is significantly different when the phone is in different locations. The problem of varying phone locations causes lower recognition accuracy. So it is necessary to propose the feature extraction algorithm which is independent of phone locations.

In this paper, we use Android phones to study activity recognition which the phone is in different locations. The activities of our research include walking, going upstairs and going downstairs. The phone's locations include upper pockets, shirt pockets, trouser pockets and hands. First, we propose an angle motion model which is used to illustrate human activities. Then, we introduce an angle feature extraction algorithm. Finally, we give our experiments. Our experiments show that the recognition accuracy improved by 2% through adding angle feature to original features. This paper is organized as follows. In Sect.2, we introduce some related work. In sect.3 we illustrate the angle motion model and the angle feature extraction algorithm. In Sect.4 we present our experiments and the results. In Sect.5, we conclude this paper in Sect.5.

2 Related Work

With the development of micro-electrical mechanical systems, the sensors are becoming smaller and smaller. So, accelerometers and gyroscopes can be embedded into mobile devices. When user carries smartphone, these sensors can obtain user's motion data and perceive surrounding information. This makes the mobile context awareness gradually become a hotspot in academia. Mobile context awareness includes activity recognition, fall detection, environment awareness and transportation mode recognition. The main steps include data collection, preprocessing, feature extraction, training and recognition. Hoseini-Tabatabaei et al [1] and Bulling et al [2] concluded related researches on mobile context awareness and activity recognition. Figo et al [3] summarized the current research on activity recognition's feature extraction. These works are instructive on activity recognition.

At the beginning of this field, researchers didn't consider the problem of varying phone locations. Kwapisz J et al [4] studies smartphone based activity recognition used Android phone. In their research, they fixed the phone in their trouser pockets, and features of three axes are extracted. Decision tree is used for classification.

The study showed that high recognition accuracy can be achieved when the phone is fixed on a specific location. However, this approach can't be applied to the problem of varying phone location.

Many motion models were proposed to illustrate human body movements, including vertical-horizontal based model [5] [6], linear-rotary based model [7] and so on. Jun Yang et al. decomposed an activity into a vertical movement and a horizontal movement to solve the orientation problem. The average acceleration of each sliding window was regard as gravity acceleration [8]. Then, resultant acceleration was decomposed into vertical and horizontal directions, and features of these two axes were extracted. Compared with extracting feature of three axes in [4], the approach was orientation independent, but the authors did not consider phone locations.

Yiqiang Chen et al [9] proposed a location independent activity recognition model. First, they extracted 145 features of resultant acceleration. The features included time domain, frequency domain, and so forth. Then, principal component analysis (PCA) was used to eliminate noise features. Finally, extreme learning machine (ELM) [10] was used to adapt the recognition model to new locations. In their research, smartphone's locations included trouser pockets, shirt pockets and hands, which involve most phone locations of daily life. However, because that the accelerations of different body parts usually varies a lot, the increase of mobile location inevitably lead to a decline in recognition accuracy. With the issue of varying phone positions, the recognition accuracy of this approach was less than 90%, and the algorithm was time-consuming. We improved feature extraction algorithms by replacing the original features [9] with FFT [11]. Our approach improved the recognition accuracy in the case of reducing time consumption. However, for the issue of phone locations are not fixed, the recognition accuracy is less than 90%. So, the feature extraction algorithm of varying phone positions should be further improved. The main purpose of this paper is to improve recognition accuracy through modifying the feature extraction algorithm.

3 Angle Motion Model

Our preliminary experiments and related studies [9][11] show that the recognition accuracy of walking, going upstairs and going downstairs is low because of their similarity. To address this issue, we propose the angle motion model and the angle feature extraction algorithm. The angle motion model is used to illustrate movement direction of walking, going upstairs and going downstairs. Because the body's movement direction is irrelevant to the changes of phone locations, adding angle feature to original features is helpful to improve recognition accuracy. This section firstly introduces the angle motion model, and then proposes an angle feature extraction algorithm.

3.1 Model Description

According to our daily experience, when user is walking on a level road, the path of user's gravity center is not a straight line which parallels to the road. The distance between body's gravity center and road periodically change in walking. Each step contains an upward motion and a downward motion. By analyzing user's movement acceleration data, we divide one step into two phases. The first phase's direction is up-forward, and the second phase's direction is down-forward. Fig. 1 shows the two phases in one step.

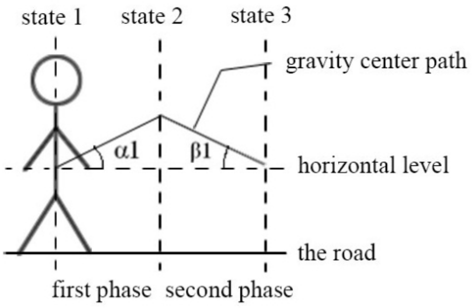


Fig. 1. The angle motion model of walking

Based on the analysis of human's movement, the body has up-forward accelerations in the first phase of one step. The accelerations rise body's gravity center and make body move forward from standing still. In Fig.1, the action between state 1 and state 2 corresponds to the first phase of one step. At the beginning of the first phase, the body is accelerating, and the direction of accelerations are up-forward. Before the body reaches the highest point, the body will decelerate, and the direction of accelerations is down-backward. Accelerations' direction of this phase is shown in Fig.2's first phase.

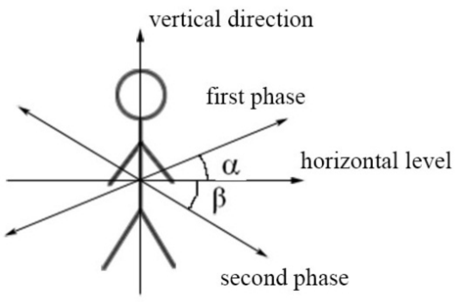


Fig. 2. The direction of accelerations

As movement continues, the body's gravity center begins to decline after achieving the highest point. At this point, the body's movement direction is down-forward which corresponds to the second phase in Fig.1. Similar to the first phase, at the beginning of the second phase, the body is accelerating, and the direction of accelerations is down-forward. Before the gravity center reaches the lowest point, the body will decelerate, and the direction of accelerations is back-upward. The second phase's accelerations direction is shown as Fig.2's second phase. In Fig.2, α correspond with first phase's angles, and β correspond with second phase's angles.

According to the same analysis, we can get the angle motion model of going upstairs, as shown in Fig.3. Comparing Fig.1 and Fig.3, we can know that the first phase's angles of the two activities are significantly different. When the user is walking on a level road, the angles between the accelerations direction and the horizontal level are generated by the body's ups and downs. So these angles are relatively small. But in the first phase of going upstairs, the angles between the accelerations direction and the horizontal level include two parts. The first parts are stairs' angles, and the second parts are angles between acceleration directions and the stairs. So, the first phase's angles of going upstairs are larger than walking's. Corresponding to the angle motion model, Fig.3's α2 are larger than Fig.1's α1.

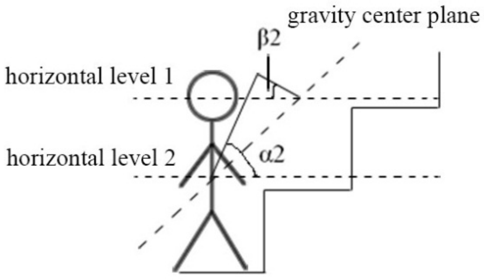


Fig. 3. The angle motion model of going upstairs

In the second phase of going upstairs, the main direction of body movement is forward. So the angles between the accelerations direction and the horizontal level are small. This means Fig.3's β2 are approximately equal to Fig.1's β1. The mean value of all accelerations' angles in one step is called mean angle in this paper. Based on the above analysis, we can get that going upstairs' mean angle is larger than walking's.

As the same analysis of going upstairs, we can get going downstairs' angle motion model. In the first phase of downstairs, the body starts to move from standing still. This process is substantially the same with the first phase of walking. During this process, body's movement direction is up-forward. The angles are shown as Fig.4's α3, which are approximately equal to Fig.1's α1. In the second phase of going downstairs, the angles between the acceleration direction and the horizontal level include two parts. The first parts are the stairs' angles, and the second parts are angles between acceleration directions and the stairs. The angles are shown as Fig.4's β3, which are larger than Fig.1's β1. Thus, going downstairs' mean angle in one step is larger than the walking's.

It can be seen from the above figures, although going upstairs and going downstairs have substantially the same angles, walking's angles are significantly different. Going upstairs and going downstairs' mean angle of one step are larger than walking's. In addition, activities' difference on mean angle is less effect by phone's locations. Therefore, adding movement angle to original features will improve recognition accuracy.

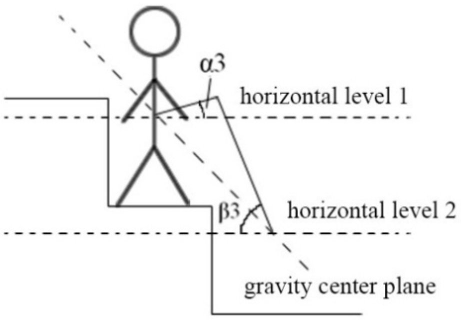


Fig. 4. The angle motion model of going downstairs

3.2 Weighted Mean Angle Feature

In this paper, half overlapping sliding window [4][5][9][11] is used to separate the collected acceleration into a number of windows. In the recognition, we assume accelerations in a window belong to the same activity. Features extracted from one window are used for classification. According to the description of section 3.1, the angles between accelerations direction and horizontal level are helpful to classify different activities. This section proposes the angle feature extraction algorithm.

Getting gravity accelerations is essential to calculating angles between the accelerations direction and the horizontal level. In Android phones, we can get gravity accelerations by calling system interface [12]. Before calculating angles, we should subtract gravity accelerations from collected accelerations. The results are actual accelerations of body movement. Then, angles are calculated by gravity accelerations and the subtracted accelerations. In this paper, gravity accelerations are written as $\boldsymbol{g}$. The subtracted accelerations are designated as $\boldsymbol{a}$. After decomposing $\boldsymbol{a}$, the horizontal components are denoted by $\boldsymbol{h}$, and the vertical components are denoted by $\boldsymbol{v}$. The horizontal level is denoted by $\boldsymbol{xy}$, and the angles between the accelerations direction and the horizontal level are written as $\boldsymbol{\theta}$. Accelerations decomposition and angles calculation are shown in Fig.5.

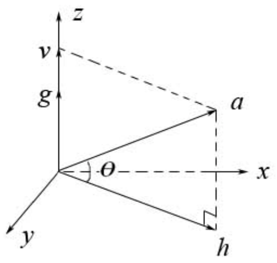


Fig. 5. Accelerations decomposition and angles calculation

For each acceleration data of a window, the angle is calculated as Formula (1).

$$\theta = \text{abs}\left(90° - \arccos\frac{\boldsymbol{a}\cdot\boldsymbol{g}}{|\boldsymbol{a}| * |\boldsymbol{g}|}\right) \tag{1}$$

Where $\boldsymbol{a}\cdot\boldsymbol{g}$ is the dot product of two vectors, and $|\boldsymbol{a}| * |\boldsymbol{g}|$ is the product of two vectors' length. As shown in the previous section, the acceleration direction is forward sometimes and backward sometimes. In practice, we can't identify if the acceleration direction is consistent with motion direction and gravity acceleration direction. So the angle between the acceleration direction and the horizontal level is of absolute value, which is calculated by the abs operator. The angle ranges from $0°$ to $90°$.

For a window which includes n accelerations, we can get n angles through Formula (1). The next step is extracting the window's angle feature through these n angles. The simplest way is to calculate the mean of these angles, and the result is regard as window's angle feature. According to our daily experience, large accelerations are obtained by strenuous exercise. Compared with small accelerations, large accelerations are more able to represent body's movement direction, and the calculated angles are more convincing in identifying body's movement direction. In order to reflect the influence of large acceleration, the weighted average method is used to calculate window's angle feature. Weight of the ith angle is calculated as Formula (2).

$$b_i = \frac{|\boldsymbol{a_i}|}{\sum_{i=1}^{n}|\boldsymbol{a_i}|} \tag{2}$$

Where b_i is the weight of the *ith* angle, $|\boldsymbol{a_i}|$ is the *ith* acceleration's length, *n* is the window size. Weighted mean angle is calculated as Formula (3).

$$\theta_{\text{mean}} = \sum_{i=1}^{n} b_i \times \theta_i \tag{3}$$

Where θ_{mean} is the weighted mean angle of a window, b_i is the weight of the *ith* angle, and θ_i is the *ith* angle. According to our experiment data, we plot the weighted mean angle's distribution of walking, going upstairs and going downstairs, as shown in Fig.6.

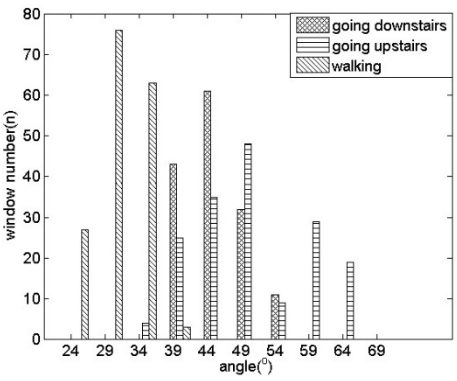


Fig. 6. The weighted mean angle's distribution of different activities

In Fig.6, the horizontal axis is angle, and the vertical axis is activities' window number in different angles. It can be seen from Fig.6 that window's distribution of three activities is significantly different. Most walking windows' weighted mean angles is less than 40, and the value of going upstairs and going downstairs is larger than 40. Therefore, adding angle feature to original features can effectively distinguish walking from going upstairs and going downstairs. Detailed experiments will be given in Sect.4.

4 Experiments

On the basis of previous studies, we use 10 lower FFT results and angle feature as activity features. ELM is used for classification. This section first introduces the collecting and processing of experiment data, and then introduces the recognition accuracy of different feature extraction algorithms. Finally the confusion matrix of different activities is given. Experiment result shows that the recognition accuracy is improved by 2% when using the proposed feature extraction algorithm.

4.1 Data Collecting

In this paper, the phone locations include chest pockets, shirt pockets, trouser pockets and hands, as shown in Fig.7. When the phone is in the hands, users swing their hands back and forth as their daily habits.

The types of phone locations in existing public activity dataset [13] are inadequate to meet our requirements. In order to validate the proposed activity recognition algorithm, we first collect the activities data of different users. The smartphone of MIUI 2s is used to collect data. This smartphone embed accelerometer, gyroscope and other commonly used sensors. These sensors can get the acceleration of smartphone's movement. In the data collecting, the sample frequency can be set to different values. According to the previous studies, sample frequency of data collecting in this paper is set to 100Hz. The activities include walking, going upstairs and going downstairs.

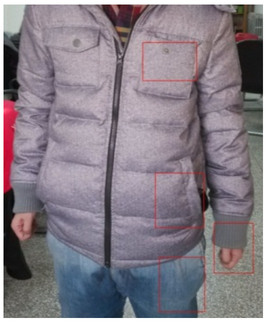


Fig. 7. The positions of data collecting

Because of actions' personalization [14][15], the feature extraction algorithm may achieve different recognition accuracies on different users' data. To exclude accident of a signal person's activity, we collected 10 persons' activity data. The participants have different ages, heights and weights, and they are not informed of the purposes and methods of experiments. Participants are asked to do different activities in their daily behavior. The collecting time for each activity at each position is approximately 1 minute. The collected data is automatically saved in the phone's storage device. After the collecting, we copy the data to PC for simulation. Features are extracted from a sliding window of 256 samples with 50% overlapping between consecutive windows. Table 1 shows the window number of each activity and each participant.

Table 1. The window number of different activities

Name	Going upstairs	Going downstairs	Walking	overall
Wch	127	138	138	403
Wcj	82	77	115	274
Zyd	109	97	121	327
Rsw	106	97	132	335
wj	110	101	129	340
Jxp	112	102	127	341
Yy	127	123	99	349
Lm	115	115	106	336
Wy	103	92	127	322
Lk	107	83	128	318
overall	1098	1025	1222	3345

4.2 Experiments Results

To avoid influence of the accident, we conducted five experiments for each participant's data. In each experiment, we divide all windows into two parts randomly.

The first part is selected to be training set, and the rest as testing set. The average of five experiments' results is as the final result of this participant.

In the training phase, window features are extracted first, and then these features are used to train recognition model. In the recognition phase, we first select a window as a coming activity. Then the window's features are extracted using the same algorithm. Finally, we use recognition model to classify this activity. We record the number of activities which is recognized correct. The ratio between correct number and total testing number is as recognition accuracy. Because of randomness of ELM's initial parameters, for the same training and testing set, we can get different recognition accuracy in each simulation. For the same training and testing set, we perform the simulation 1000 times to avoid the influence of ELM's randomness. Average result of these 1000 simulations is as the final recognition accuracy.

Table 2 shows all participants' recognition accuracy of different feature extraction algorithms. PCA145 is the algorithm of [9], FFT10 is our previous algorithm in [11], and AFFT10 is the proposed algorithm in this paper which is adding angle feature to FFT10.

Table 2. The recognition accuracy of different algorithms

Name	PCA145(%)	FFT10(%)	AFFT10(%)
Wch	85.74	86.82	90.44
Wcj	92.13	91.17	93.46
Zyd	88.78	89.51	91.31
Rsw	84.86	86.15	90.06
Wj	93.61	94.15	94.99
Jxp	90.93	91.38	95.69
Yy	89.70	85.61	87.10
Lm	82.02	84.35	85.68
Wy	91.36	91.57	92.06
Lk	94.52	94.13	95.32
average	89.37	89.48	91.61

As is shown in Table 2, the recognition accuracies are different from person to person for the same feature extraction and classification algorithm. When the feature extraction algorithm is PCA145, the highest recognition accuracy is 94.52%, the lowest recognition accuracy is 82.02%, and the average recognition accuracy is 89.28%. When the feature extraction algorithm is FFT10, the average recognition accuracy is substantially equal with those of PCA145. Compared with the PCA145 and FFT10, AFFT10 improves the average recognition accuracy by 2.31%, and each participant's recognition accuracy has risen of certain level.

Table 3. The confusion matrix of FFT10

	Going upstairs	Going downstairs	Walking
Going upstairs	85.60%	5.26%	9.14%
Going downstairs	9.35%	84.62%	6.03%
Walking	1.23%	1.94%	96.83%

Table 4. The confusion matrix of AFFT10

	Going upstairs	Going downstairs	Walking
Going upstairs	87.81%	4.91%	7.28%
Going downstairs	9.98%	85.09%	4.93%
Walking	1.23%	1.09%	97.68%

For a detailed analysis of our experimental results, we record the confusion matrix of different activities. The confusion matrix is shown as Table 3 and Table 4.In Table 3 and Table 4, the value in the ith row and jth column means the percentage which the ith activity is recognized as the jth. Take Table 3's first line for example, the recognition accuracy of going upstairs is 85.60% when using the FFT10 algorithm. For going upstairs, 5.26% of the windows are recognized as going downstairs, and 9.14% of the windows are recognized as walking.

Comparing Table 3 with Table 4, we can find that the confusion between going upstairs and going downstairs is not significantly changed after using our algorithm. The confusion between these two activities is maintaining at around 5% and 9% respectively. But the confusion between walking and these two activities is significantly declining. The percentage which going upstairs recognized as walking reduces from 9.14% to 7.28%, and the percentage which downstairs recognized as walking reduces from 6.03% to 4.93%. The recognition confusion decreases by about 2% and 1% respectively. By analyzing the confusion matrix, we can see that adding the angle feature to original features can reduce the confusion between walking and going upstairs/downstairs. Thus, the overall recognition accuracy is improved.

In our experiment, the duration of a window is about 1.28 second. Our experiment shows that our method's consuming time is less than 200ms. So, there is enough time to recognition an activity.

5 Conclusion

Compared with traditional wearable computing which fixes accelerometers on specific positions of user's body, activity recognition based on smartphone faces a new problem of varying phone location. In this paper, Android phone is used to study activity recognition which the phone is not fixed. The main work of this paper is proposing the position independent feature extraction algorithm. According to the law of human's activity, we first establish the angle motion model, and point out activities' difference in acceleration angle. Then, we propose the angle feature extraction algorithm. Analysis of our experiment data shows that the angle feature is less influenced by phone locations. Finally, this paper describes our experiments, including the data collecting, the recognition accuracy and the confusion matrix of different algorithms.

In our experiment, smartphone of MIUI 2S is used to collect data. The activities include walking, going upstairs and going downstairs. The phone's locations include chest pockets, shirt pockets, trouser pockets and hands. Ten users' activities data are used for simulation. Compared with the previous studies, our algorithm improves the recognition accuracy by 2%.

There are many problems worth studying in smartphone based activity recognition. In this paper, we just use the absolute angle to calculate the angle feature. We haven't considered the relationship between the angles and the movement direction. In the further work, adding the movement direction to the angle feature extraction algorithm is helpful to improve the recognition accuracy. In addition, the activity pattern of a specific user may change at different time of a day. Motion-adaptive activity recognition is a problem which should be addressed in the future work.

Acknowledgements. This work was supported by the Research Fund for the Doctoral Program of Higher Education of China (No.20110031110026) and (No. 20120031110035) and the Fundamental Research Funds for the Central Universities - the Science Foundation for Young Scholars of Nankai University (No. z1a2035946).

References

1. Hoseini, S.A., Gluhak, A., Tafazolli, R.: A survey on smartphone-based systems for opportunistic user context recognition. ACM Computing Surveys (CSUR) 45, 1–55 (2013)
2. Bulling, A., Blanke, U., Schiele, B.: A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys (CSUR) 46, 1–33 (2014)
3. Figo, D., Diniz, P.C., Ferreira, D.R., Cardoso, J.M.: Preprocessing techniques for context recognition from accelerometer data. Personal and Ubiquitous Computing 14, 645–662 (2010)
4. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SIGKDD Explorations Newsletter 12, 74–82 (2011)
5. Yang, J.: Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In: Proceedings of the 1st International Workshop on Interactive Multimedia for Consumer Electronics, pp.1–10. ACM Press, New York (2009)
6. Park, J., Patel, A., Curtis, D., Teller, S., Ledlie, J.: Online pose classification and walking speed estimation using handheld devices. In: Proceedings of the 14th International Conference on Ubiquitous Computing, pp. 113–122. ACM Press, New (2012)
7. Shi, Y., Shi, Y., Liu, J.: A rotation based method for detecting on-body positions of mobile devices. In: Proceedings of the 13th International Conference on Ubiquitous Computing, pp. 559–560. ACM Press, BeiJing (2011)
8. Mizell, D.: Using gravity to estimate accelerometer orientation. In: Proceedings of the 7th IEEE International Symposium on Wearable Computers, pp. 252–253. IEEE Press, New York (2003)

9. Chen, Y., Zhao, Z., Wang, S., Chen, Z.: Extreme learning machine-based device displacement free activity recognition model. Soft Computing 16, 1617–1625 (2012)
10. Huang, G.B., Wang, D.H., Lan, Y.: Extreme learning machines: A survey. International Journal of Machine Learning and Cybernetics 2, 107–122 (2011)
11. Wang, C., Zhang, J.Z., Wang, Z.C., Wang, J.: Position-independent activity recognition model for smartphone based on frequency domain algorithm. In: The 3rd International Conference on Computer Science and Network Technology, pp. 305–308. IEEE Press, Da Lian (2013)
12. Kourogi, M., Kurata, T.: Personal positioning based on walking locomotion analysis with self-contained sensors and a wearable camera. In: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, pp. 103–107. IEEE Press, Tokyo (2003)
13. Xue, Y., Jin, L.: A naturalistic 3D acceleration-based activity dataset & benchmark evaluations. In: IEEE International Conference on Systems Man and Cybernetics (SMC), pp. 4081–4085. IEEE Press, Barcelona (2010)
14. Weiss, G.M., Lockhart, J.W.: The impact of personalization on Smartphone-based activity recognition. Technical report, Workshop on Activity Context Representation (2012)
15. Zhao, Z., Chen, Y., Liu, J., Shen, Z.Q., Liu, M.: Cross-people mobile-phone based activity recognition. In: Proceedings of the 22th International Joint Conference on Artificial Intelligence, pp. 2545–2550. AAAI Press, Barcelona (2011)

Reliable and Energy Efficient Routing Algorithm for WirelessHART

Qun Zhang, Feng Li, Lei Ju, Zhiping Jia, and Zhaopeng Zhang

School of Computer Science and Technology, Shandong University, Ji'nan, China, 250101
lifeng6699@sdu.edu.cn

Abstract. WirelessHART is the first open wireless communication standard designed for industrial monitoring and control. Process industry applications for real-time plant have stringent needs of reliability, stability and security of wireless communication. In this paper we propose an approach for reliable routing graph construction and energy efficient link selection to enhance the reliability of the communication and prolong the network lifetime. In particular, we ensure at least two neighbor nodes are maintained for each node to support increase the delivery ratio of hop-level retransmission. Furthermore, link quality and energy model are considered in the link selection process, which reduces the potential number of retransmission and balances the residual energy in the network. Experimental results show that our proposed mechanism outperforms the state-of-the-art WirelessHART routing algorithms in terms of extending the network lifetime and the reliability of message transmission.

1 Introduction

WirelessHART is the first open-standard wireless communication protocol designed for industrial automation and process [1]. The WirelessHART standard uses TDMA and channel hopping techniques to control access of the network and to communicate between network devices. WirelessHART advocates explicit and centralized network management for reliable and real-time network communication. Before WirelessHART is released, there have been a few publicly available standards on office automation, such as ZigBee[2] and Bluetooth[3]. However, these technologies cannot meet the stringent requirements of industrial applications [1]. Different from the star network or the tree network, the topology of WirelessHART is a mesh network in which data can be transmitted (or retransmitted) along different paths. Single path routing algorithms focus on picking up an energy efficiency path to prolong the lifetime of the network [4]. Multipath routing takes advantage of interconnect nodes network to enhance data throughput and packet delivery ratio [5]. Hop-based routing protocol has been receiving extensive attention for its simple and effective design ideas.

In WirelessHART network, routing plays very important role for its reliability requirement and limited communication resources. [9,10,11] introduce algorithms to construct routing graph, and [16,17,18] describe link selection methods considering energy efficient. But a routing algorithm with both appropriate routing graph and optimal link selection is more feasible for the industrial environment.

In this work, we propose an approach for reliable routing graph construction and energy efficient link selection to enhance the robustness and prolong the network lifetime. In routing graph construction, we re-add a link to guarantee sufficient choice for link selection. When selecting links, we mainly consider link quality, energy efficient and the number of hops to choose optimal links. We use the analytic hierarchy process to determine the weight coefficients. Redundant links and the link selection can increase the successful rate of the transmission and decline the number of retransmission. In the simulation, we have compared our algorithm with two existing WirelessHART routing protocols JRMNL[17] and RUG[11]. Experimental results show that our proposed algorithm outperforms in terms of extending the network lifetime and reliability of message transmission.

2 Related Work

In recent years, there have been a lot of researches about wireless sensor network routing algorithm [4]. SMR is designed to utilize multipath concurrently by splitting traffic onto two maximally disjoint routes [6]. Ganesan et al. proposed a node-disjoint and braided multipath scheme to provide energy efficiency and resilience against node failures [7]. Ye et al. proposed an algorithm AODVM which is an extension to AODV for finding multiple node-disjoint paths [8].

According to the specific routing mechanism in WirelessHART, Zhao et al. proposed a routing algorithm based on least-hop called ELHFR [9]. Gao et al. proposed a multipath graph routing algorithm with subgraph called ORMGR [10]. Song et al. proposed algorithms to construct three types of reliable routing graphs for different communication purposes [11].

However, in the above-mentioned work, least-hop is the only criterion when choosing the links. In practice, devices are equipped with battery and the energy resources are constrained. We should choose optimal links to balance the residual energy such that each node has routing responsibility in a fair way.

Hung et al. considered the robustness of the path connection, and an energy-efficient opportunistic routing strategy EFFORT was proposed [5]. Liu et al. proposed a clustering algorithm based on time delay and energy awareness [12, 13]. Karbaschi et al. proposed a routing algorithm judging the link quality judgment and prolonging the network lifetime [14]. Ibrahim et al. proposed the Minimum Transmission Power Cooperative Routing algorithm (MPCR) [15]. It can reduce the energy consumption of a single route while guarantee certain throughput.

As we can see, these algorithms are applied to the distributed wireless sensor network, but the routing in WirelessHART is managed by the centralized network manager. In WirelessHART, Wang et al. proposed a routing algorithm called DHEIRP, by which a node choose the optimal next hop through comparing the residual energy of neighbor nodes [16]. The algorithm in [17], JRMNL, chooses the next hop according to node communication load, node residual energy and link transmission energy consumption. In addition, in [18], a load-balanced routing algorithm based on communication frequency proposed that a node always chooses the next-hop node by comparing the neighbor communication load.

In practice, there are some low quality links which may increase the communication failure probability and will further cause waste of energy. Based on the graph routing, this paper proposes a method of constructing the topology graph with redundant links and an algorithm about links selection by taking into account the remaining energy, the quality of the link and the number of hops. Redundant links can increase the reliability and robustness of the network, and provide sufficient choices for link selection. The link selection can increase the successful rate of the transmission and decline the potential number of retransmission. The residual energy will be balanced and the network lifetime can be prolonged to meet the strict requirements in industrial application.

3 System Model

3.1 WirelessHART Network

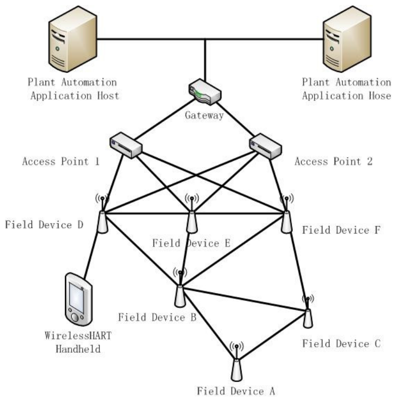


Fig. 1. WirelessHART mesh networking

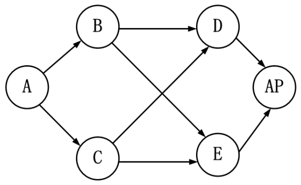
As shown in Fig. 1, a WirelessHART mesh network consists of different kinds of devices. All of them can send and receive datagram and perform some basic function to support the formation and maintenance of the network.

3.2 Routing Mechanisms

There are two routing mechanisms defined in WirelessHART [3]. Source routing is a one-way path between the source device and destination device, which is mostly used for network diagnostics. In graph routing, when sending a packet, the source device determines the specific *Graph_ID* according to the routing table, and writes it in NPDU(network protocol data unit). The intermediate nodes just need to check

Graph_ID after receiving the packet, and choose the next neighbor in the relevant *ID*. In this paper, our study is based on graph routing which is the main routing protocol in WirelessHART.

3.3 Superframe Schedule



(a) Example of topology graph

Time Slot	TS0	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Channel0	A→B		A→C	B→D	B→E	C→E	D→AP	E→AP
Channel1		A→B	B→D	C→E	C→D	D→AP		

(b) Example of superframe

Fig. 2. Example of a graph route-based superframe

Based on TDMA, WirelessHART uses 10ms time-slots, in which the transmission of a packet from a node to its neighbor must be accomplished. A superframe is composed of a number of time-slots. Dang proposed a graph route-based superframe scheduling scheme [19]. According to the strategy, the superframe of the graph shown in Fig. 2(a) can be illustrated in Fig. 2(b).

4 The Re-add Routing Algorithm

4.1 Network Model

According to WirelessHART standard, we suppose the radiation radius of all nodes is *R*. When nodes are defined as vertices and links are defined as edges, the network can be defined as a simple undirected graph *G=(V,E)*, where *V* denotes the set of vertices and *E* denotes the set of edges. There is an undirected link *w(i, j)* connecting two nodes *i* and *j* when the two nodes are within each other's transmission range. We use *g* to denote the Gateway, V_{AP} to denote the set of Access Points and V_D to denote the set of devices. We have $\{g\} \cup V_{AP} \cup V_D = V$.

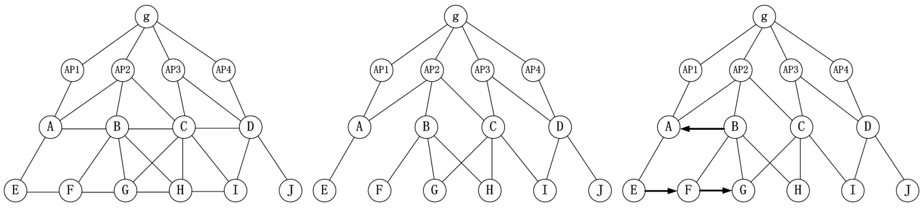
4.2 Topology Construction

The following is our process of forming a network, and here we only focus on the uplink graph, with downlink similar to this.

Firstly, we initialize the network manager to create and initialize the gateway, and set the degree of the gateway to be 0. When a new node requests to join the network, all nodes which are within the radiation radius *R* can be recorded as its neighbors. If the minimum degree of the neighbor nodes is *i*, the degree of the new node is *i*+1. The network topology construction finishes until all nodes form links with other nodes and have their degrees. In the network graph shown in Fig. 3(a), the degree of node *g* is 0; the degree of node *AP1*, *AP2*, *AP3* and *AP4* is 1; the degree of node *A*, *B*, *C* and *D* is 2; the degree of node *E, F, G, H, I* and *J* is 3.

Industrial networks have stringent real time requirements, so minimum possible paths should be preferred. After all nodes' degrees are marked, remove the links between nodes with the same degrees and establish the simplified topology with the shortest path to the gateway. In Fig. 3(a), the links between *AB, BC, CD, EF, FG, GH* and *HI* will be removed. The simplification graph is shown in Fig. 3(b).

However, the simplification may cause some nodes lose their redundant links, and only have one neighbor to transmit. To enhance the route reliability, we re- add a link for these nodes. The re-added link is obviously between the node and its neighbor with same degree, and we choose the shortest link to ensure the least transmission energy. In Fig. 3(b), the node *F* only has one neighbor *B* to transmit. As *G* is closer to *F*, we re-add the link from *F* to *G*. According to the algorithm, we also re-add links for *B* and *E*. The Fig. 3(c) is the final topology graph.



a) Origin topology (b) Simplification graph (c) Re-add graph

Fig. 3. Graph routing topology structure

```
Algorithm 1 Constructing Uplink Graph G'(V',E')
Input:
G(V,E): the origin graph
$h_{v'}$: the minimum number of hops of node v'
$d_{v',u'}$: the distance between node v' and node u'
Output:
G'(V',E'): the Re-add graph
1: G'(V',E')=G(V,E)
2: for all node v' ∈ V' do
3:     for all edges $e_{v',u'}$ from E' about v' do
```

```
4:            if($h_{v'} == h_{u'}$)
5:                Delete $e_{v',u'}$ from $E'$
6:            end if
7:        end for
8: end for
9: Find $S' \subseteq V'$ and $\forall v' \in S'$, $v'$ has only one neighbor to transmit
10: if $S' \neq \varnothing$ then
11:       for all node $v' \in S'$ do
12:           Sort its edges $e_{v',u'}$ according to $d_{v',u'}$ and $h_{v'} = h_{u'}$
13:           Choose the first edge and add it to $E'$
14:       end for
15: end if
```

4.3 Link Selection

This part will introduce out link selection approach. The residual energy and the energy consumption for transmission are obviously two determinants, and we also consider the link quality to enhance the reliability of transmission.

The network manager maintains the following information :

1. Degree:

During the construction of the routing graph introduced in Section4.2, every node *i* can know the minimum hop as its degree D_i .

2. Residual Energy:

We assume that *N* nodes in a WirelessHART network have the same initial energy E_0 , and node *i* knows very well about its real-time residual energy E_i . Each node can also know all its neighbors' location and residual energy.

3. Link Quality:

Link quality, denoted by *Q*, means the ratio of the number of packets received to the number of packets transmitted. The low quality link can result the decline of efficiency of the shortest path and incur significant energy expenditure due to retransmission [20]. To reduce the cost of the control message for link evaluation, Woo et al. proposed an estimator, window mean with exponentially weighted moving average (WMEWMA) [21]. The link quality can be expressed as

$$\widehat{Q}_{k+1} = m\widehat{Q}_k + (1-m)\frac{received}{received + failed}, \tag{1}$$

where $\widehat{Q}_k$ and $\widehat{Q}_{k+1}$ respectively denote the link quality evaluation at time *k* and *k*+1, *received* means the number of packets received in *t* (which is the time window represented in number of message opportunities), *failed* means the number of packets lost in *t*, and $m \in [0,1]$ controls the history of the estimator.

Information about the residual energy and the number of packets sent and received change with time. The devices will deliver a health report like Fig. 4 periodically to the network manager to update the information. After sending the health report, the device will reset the number of packets sent and received.

Device ID	Neighbor_ID	Number of packets sent to the neighbor	Number of packets received from the neighbor	……

Fig. 4. The format of the health report

4. Energy Consumption of Every Transmission：

We use d_{ij} to denote the linear distance between nodes *i* and *j*. In [22], it was proved that the energy consumption for transmission from node *i* to node *j* can be expressed as

$$E_{ij} = \frac{\left(2^{R_0} - 1\right) N_0 d_{ij}^{\alpha}}{-\log\left(p_0^s\right)}, \tag{2}$$

where α indicates the path loss exponent and R_0 and p_0^s denote the transmission rate and the desired probability of successful transmission, respectively.

5. Link Selection Priority

Based on the above information, we define the priority of the link between node *i* and node *j* as P_{ij}. And it can be estimated by

$$P_{ij} = x_1 \cdot Q_{ij} + x_2 \cdot \frac{E_i + E_j - E_{ij}}{2 \cdot E_0} + x_3 \cdot \frac{D_i - D_j}{D_i}, \tag{3}$$

where x_1 , x_2 and x_3 are weight coefficients. The three parameters are all vary between 0 and 1. The priority increases with all of them. The combination of the three parameters can well balance their importance in routing. The determination of the weight coefficients is based on the analytic hierarchy process [23]. Firstly, we construct the comparison matrix

$$A=\left(a_{ij}\right)=\left(\begin{array}{ccccc} A & x_1 & x_2 & x_3 & r_i \\ x_1 & 1 & 0 & 2 & 3 \\ x_2 & 2 & 1 & 2 & 5 \\ x_3 & 0 & 0 & 1 & 1 \end{array}\right), \tag{4}$$

where $a_{ij}=\begin{cases} 2 & x_i \text{ is more important than } x_j \\ 1 \ (i,j\text{:} & x_i \text{ is as important as } x_j \\ 0 & x_j \text{ is more important than } x_i . \end{cases}$

Then we construct the judgment matrix

$$X=\left(x_{ij}\right)=\left(\begin{array}{ccccccc} x & x_1 & x_2 & x_3 & M_i & W_i & \overline{W_i} \\ x_1 & 1 & 0.33 & 3 & 1 & 1 & 0.29 \\ x_2 & 3 & 1 & 9 & 27 & 1.93 & 0.56 \\ x_3 & 0.33 & 0.11 & 1 & 0.04 & 0.52 & 0.15 \end{array}\right), \tag{5}$$

where $x_{ij}=c_b^{\left(r_i-r_j\right)/R}$, $M_i=\prod_{j=1}^{3}x_{ij}$, $W_i=\sqrt[3]{M_i}$, $\overline{W_i}=\frac{W_i}{\sum_{i=1}^{3}W_i}$. c_b is a constant which is always assumed as 9. $R=r_{\max}-r_{\min}$ is called range, and $r_{\max}=\max\left\{r_1,r_2,r_3\right\}$, $r_{\min}=\min\left\{r_1,r_2,r_3\right\}$. Through the consistency check, we get the weight coefficients $\left(x_1,x_2,x_3\right)=\left(0.29,0.56,0.15\right)$.

By periodically collecting the routing graph, the residual energy and the link quality, the network manager can calculate the priorities of the links and update the link selection. When a node sends a packet, the link with the highest P_{ij} is selected preferentially for the transmission. In some cases, a link with more hops may be chosen. But the high link quality and more residual energy can guarantee higher success possibility of transmission to avoid the retransmissions and balance the energy in the network.

5 Experimental Result

In the simulations, we use the superframe schedule introduced in Section3.3. The detailed simulation parameters are listed in Table 1. We compare JRMNL[17], RUG[11] and our algorithm under the simulation environment.

The first experiment compares the edges in the graph by these three approaches. Fig. 6 shows that there are more edges in the graph constructed by our approaches. With retransmission mechanism and the channel hopping technology in the schedule, the advantage of the redundancy will be highlighted.

In the second experiment, we measure the number of transmissions and average residual energy when the first dead node appears without considering the retransmission. We also measure the number of transmissions and the number of packets received by the gateway when no more packets can get to the gateway. Fig. 6 and Fig. 7 record the results. The results show that our approach can make more transmissions and substantially extend the network lifetime longer.

Table 1. Simulation parameters

Square area	450m×450m
Communication radius	100m
Number of nodes	50,75,100,125,150
Initial energy E_0	1
Noise variance N_0	-70dBm
Path loss exponent α	2
Transmission rate R_0	2 b/s/Hz
Desired probability of successful transmission p_0^s	0.95

In the third experiment, we evaluate the average latency. As shown in Fig. 8, when the number of nodes is 75, the average latency in our algorithm is around 4.4% higher than JRMNL. That means the latency brought by the re-add links in our algorithm can be tolerated.

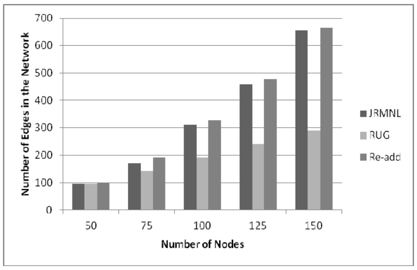
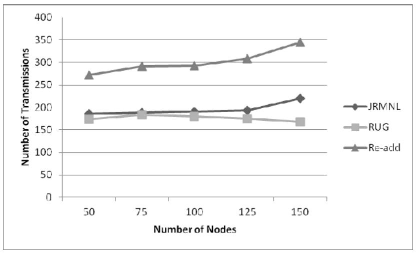
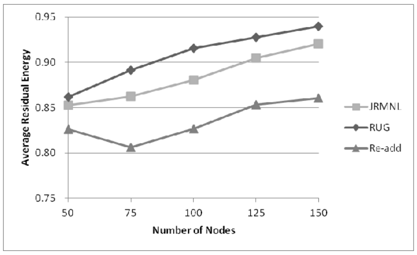


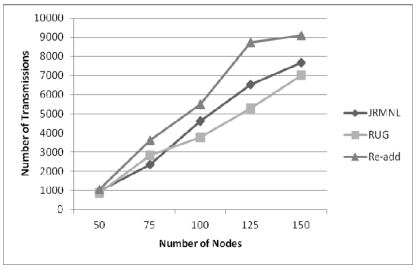
Fig. 5. Number of edges in the network



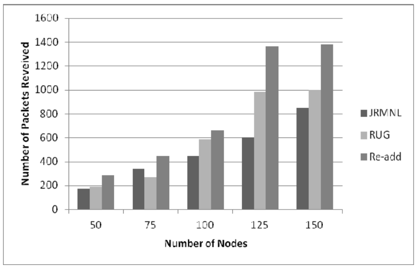


(a) Number of successful transmissions **(b)** Average residual energy

Fig. 6. Lifetime of the network (when the first dead node appears)



(**a**) Number of successful transmissions



(**b**) Number of packets received

Fig. 7. Lifetime of the network (when the gateway can't receive packets)

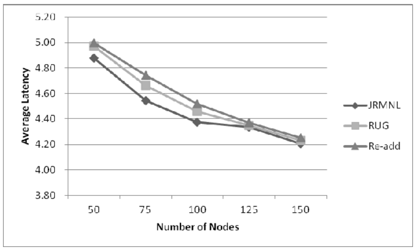


Fig. 8. Average latency

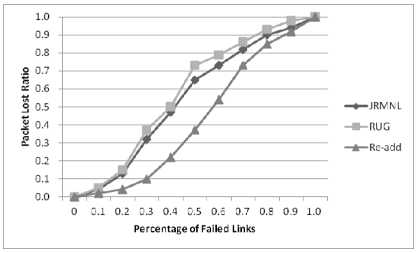


Fig. 9. Packet lost ratio

The fourth experiment records the packet lost ratio with the three algorithms. Fig. 9 expresses our algorithm has lower packet lost ratio. When the percentage of failed links is 50%, the packet lost ratio in our algorithm is around 48% lower than RUG. It states the link selection can increase the successful rate of transmission and reduce the number of retransmission.

Considering both reliability and lifetime of the network, our approach shows an obvious improvement.

6 Conclusion

Industrial wireless network needs more reliability than other wireless sensor networks because of the rigid environment. Our approach is a multilink-based graph routing protocol and selects links with higher priority according to energy efficient, quality of links and the number of hops. The graph construction and link selection algorithm are described in detail. Redundant links and link selection can increase the successful rate of the transmission and decline the number of retransmission for energy saving. Simulations show that our approach is more reliable, and prolongs the lifetime of network. It is feasible for industrial wireless network. In the future work, we shall continue to further study the superframe schedule to decrease the network delay.

Acknowledgement. This research is sponsored by the Natural Science Foundation of China (NSFC) under grant No. 61202015, Shandong Provincial Natural Science Foundation No. ZR2013FM028 and ZR2011FQ036, Research Fund for the Doctoral Program of Higher Education of China (RFDP) grant No. 20120131120033.

References

1. Song, J., Han, S., Mok, A.K., Chen, D., Lucas, M., Nixon, M., Pratt, W.: WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control. In: IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 377–386. IEEE Press (2008)
2. Karenos, K., Kalogeraki, V., Krishnamurthy, S.V.: A rate control framework for supporting multiple classes of traffic in sensor networks. In: 26th IEEE International Symposium on Real-Time Systems Symposium, pp. 287–297. IEEE Press, Miami (2005)
3. WirelessHART specification, `http://www.hartcomm2.org`
4. Pantazis, N.A., Nikolidakis, S.A., Vergados, D.D.: Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey. IEEE Communications Surveys & Tutorials 15(2), 551–591 (2013)
5. Hung, M.C., Lin, K.C., Chou, C., Hsu, C.: EFFORT: Energy-efficient opportunistic routing technology in wireless sensor networks. In: Wireless Communications and Mobile Computing, vol. 13(8), pp. 760–773 (2013)
6. Lee, S.-J., Gerla, M.: Split multipath routing with maximally disjoint paths in ad hoc networks. In: IEEE International Conference on Communications, vol. 10, pp. 3201–3205. IEEE Press, Helsinki (2001)
7. Ganesan, D., Govindan, R., Shenker, S., Estrin, D.: Highly-resilient, energy-efficient multipath routing in wireless sensor networks. ACM SIGMOBILE Mobile Computing and Communications Review 5, 11–25 (2001)
8. Ye, Z., Krishnamurthy, S.V., Tripathi, S.K.: A framework for reliable routing in mobile ad hoc networks. In: 22nd Annual Joint Conference of the IEEE Computer and Communications, vol. 1, pp. 270–280. IEEE Press (2003)
9. Zhao, J., Liang, Z., Zhao, Y.: ELHFR: A Graph Routing in Industrial Wireless Mesh Network. In: International Conference on Information and Automation, pp. 106–110. IEEE Press, Zhuhai (2009)
10. Gao, G., Zhang, H., Li, L.: A Reliable Multipath Routing Strategy for WirelessHART Mesh Networks Using Subgraph Routing. Journal of Computational Information Systems 9, 2001–2008 (2013)
11. Song, H., Zhu, X., Mok, A.K., Chen, D.: Reliable and Real-time Communication in Industrial Wireless Mesh Networks. In: 17th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 3–12. IEEE Press, Chicago (2011)
12. Shang, F.: A multi-hop routing algorithm based on integrated metrics for wireless sensor networks. Applied Mathematics & Information Sciences 7, 1021–1034 (2013)
13. Liu, M., Cao, J., Chen, G., Wang, X.: An energy-aware routing protocol in wireless sensor networks. Sensors 9, 445–462 (2009)
14. Karbaschi, G., Fladenmuller, A., Baey, S.: On the link-quality aware routing in wireless multi-hop networks: A throughtput-stability trade-off study. In: IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5. IEEE Press, Athens (2007)
15. Ibrahim, A., Zhu, H., Liu, K.J.R.: Distributed energy-efficient cooperative routing in wireless networks. IEEE Transactions on Wireless Communications 7(10), 3930–3941 (2008)
16. Wang, Y., Zhang, S., Lin, X.: Distributed low-power dissipation routing algorithm based on WirelessHART. Modern Electronics Technique, 60–64 (2013)
17. Zhang, S., Yan, A., Ma, T.: Energy-Balanced Routing for Maximizing Network Lifetime in WirelessHART. International Journal of Distributed Sensor Networks, 1–8 (2013)

18. Memon, A.A., Hong, S.H.: Minimum-Hop Load-Balancing Graph Routing Algorithm for Wireless HART. International Journal of Information and Electronics Engineering 3, 221–225 (2013)
19. Dang, K., Shen, J.Z., Dong, L.D., Xia, Y.X.: A Graph Route-Based Superframe Scheduling Scheme. Wireless Personal Communications 71(4), 2431–2444 (2013)
20. Zhao, J., Govindan, R.: Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In: The 1st International Conference on Embedded Networked Sensor Systems, pp. 1–13. ACM, New York (2003)
21. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multi-hop routing in sensor networks. In: The 1st International Conference on Embedded Networked Sensor Systems, pp. 14–27. ACM, New York (2003)
22. Ibrahim, A.S., Zhu, H., Liu, K.J.R.: Distributed Energy-Efficient Cooperative Routing in Wireless Networks. IEEE Transactions on Wireless Communication, 3930–3941 (2008)
23. Saaty, T.L.: Analytic hierarchy process. In: Encyclopedia of Operations Research and Management Science, pp. 52–64. Springer, US (2013)

A DSCP-Based Method of QoS Class Mapping between WLAN and EPS Network

Yao Liu, Gang Lu, Wei Zhang, Fengling Cai, and Qian Kong

School of Information Science and Technology, East China Normal University
500 Dongchuan Rd., Shanghai, China
liuyao@cc.ecnu.edu.cn, {glu,wzhang}@cs.ecnu.edu.cn,
{flcai,kqian}@student.ecnu.edu.cn

Abstract. In recent years the developments of wireless and mobile networks are particularly prominent. In order to provide users with better services over these heterogeneous networks, QoS needs to be considered. In this paper, we analyze the QoS features of LTE and WLAN respectively. Then we propose a DSCP-based method of QoS class mapping between WLAN and EPS network, and simulate the QoS class mapping method in OPNET. By analyzing the experiments, we consider that this method is feasible and effective. Finally we present the conclusion.

Keywords: QoS, Class Mapping, EPS, WLAN, QCI, 802.11ç, EDCA, DSCP.

1 Introduction

The trend of forthcoming Next Generation Network (NGN) is an all-IP network over heterogeneous networks. The networks are composed of different network technologies, such as IEEE 802.11 WLAN, IEEE 802.3 Ethernet, 3GPP EPS, traditional Internet, etc.

Among these network technologies, the developments of wireless and mobile networks are particularly prominent in recent years. The mobile communication technology has entered the era of 4G, which can provide conversational and multimedia services, with the advantages of large coverage and global roaming. WLAN can provide high data transmission rate, with the advantages of flexibility and low cost.

In order to improve the quality of service (QoS) provisioning, IEEE has specified the 802.11e protocol [1], which provides the mechanisms to support the applications with QoS requirements. 3GPP also has standardized the PCC architecture [2], which supports the fine-grained QoS and dynamic control of the QoS and charging requirements for IP Multimedia Subsystem (IMS) and non-IMS services.

Good service quality is an important factor for users, so QoS is one of the important problems to be solved in the heterogeneous networks. However, different networks have different and probably incompatible QoS provisioning methods.

There are many studies that considered the end-to-end QoS provisioning over heterogeneous networks. In general, the previous studies can be categorized into two approaches: mapping between QoS parameters or mapping between QoS classes.

The QoS parameter mapping converts the QoS parameters among the heterogeneous networks. The QoS class mapping converts the QoS service classes which are defined for each service requirement among the heterogeneous networks.

The AQCM-ASM framework to support class mapping and fine-granular QoS over the heterogeneous networks was proposed in [3]. The QoS class mapping for heterogeneous QoS domains using flow aggregation was discussed in [4]. The QoS mapping between WLAN and UMTS networks was considered in [5]. The mapping of QoS classes between UMTS, WiMAX and DiffServ/MPLS networks was discussed in [6]. An end-to-end LTE QoS signaling protocol was presented in [7], which supported LTE QoS model mapping to QSPEC objects. A QoS mapping scheme for MPLS-DiffServ and label forwarding was proposed in [8]. The method of mapping the LTE QoS parameters was discussed in [9]. And we proposed a QoS class mapping in WLAN (using HCCA mechanism) and 3GPP LTE interworking network [10]. However, few studies discussed the complete method of QoS class mapping between WLAN and EPS network.

The goal of this paper is to provide a DSCP-based method of QoS class mapping between WLAN and EPS network, which are interworked with Internet. For this study, we analyze the QoS features of LTE and WLAN respectively. Then we propose a dedicated method of QoS class mapping, which is based on DSCP, between WLAN and EPS network, and simulate and analyze the QoS class mapping method in OPNET. Finally we present the conclusions.

2 QoS Mechanisms of EPS and WLAN

2.1 QoS Mechanisms of EPS

In order to provide the dynamic service-aware QoS and charging control, the Policy and Charging Control (PCC) was proposed and included in EPS [2]. The PCC architecture focused on QoS is composed of PCRF, PCEF, BBERF, SPR/UDR, TDF and AF, shown in Fig. 1.

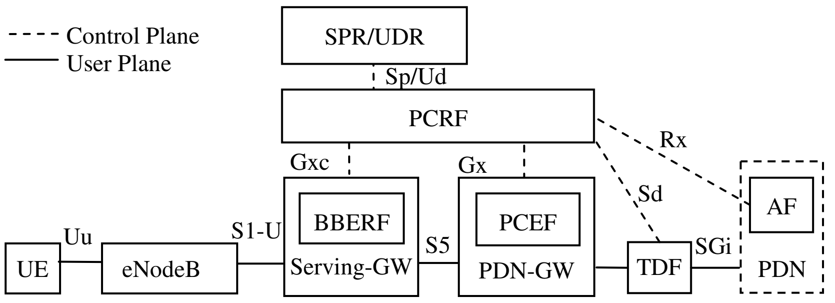


Fig. 1. The PCC architecture focused on QoS

- The Policy and Charging Rule Function (PCRF) is the central entity in PCC. It makes the PCC decisions (PCC rules) based on many information received from

PCEF, BBERF (if used) and SPR, and operator configuration in the PCRF. The PCC rules will be sent to the PCEF, and the subset of PCC rules (QoS rules) will be sent to the BBERF if the off-path model is used.

- The Policy and Charging Enforcement Function (PCEF) located in the PDN-GW receives PCC rules from PCRF, and enforces the decisions. Meanwhile, the user-specific and access-specific information will be provided to the PCRF.
- The Bearer Binding and Event Reporting Function (BBERF) located in the access GW if the off-path model is used, provides a subset of the functionalities of the PCEF.
- The Subscription Profile Repository (SPR) is the PCC-related subscription database, and the User Data Repository (UDR) is another alternative database.
- The Application Function (AF) is an application entity, which corresponds to the P-CSCF for IMS or an application server (e.g., a streaming server) for a non-IMS service. Its applications or services need dynamic PCC. The AF provides session information extracted from application signaling to the PCRF. Meanwhile, it receives the PCC-related events at traffic-plane level.
- The Traffic Detection Function (TDF) is an application detection entity, which uses deep packet inspection and reports the results of detection to the PCRF.

In Evolved Packet System (EPS), every EPS bearer is assigned to a QoS Class Identifier (QCI) which is a numerical scalar (1-9). The QCI is used to represent the packet forwarding behavior. 3GPP defined the QCI characteristics [2], and [11] described it in more detail. The standardized QCI characteristics are shown in Table 1.

Table 1. Standardized QCI characteristics

QCI	Resource Type	Priority	Packet Delay Budget	Packet Error Loss Rate	Example Services
1	GBR	2	100 ms	10^{-2}	Conversational Voice
2		4	150 ms	10^{-3}	Conversational Video (Live Streaming)
3		3	50 ms	10^{-3}	Real Time Gaming
4		5	300 ms	10^{-6}	Non-Conversational Video (Buffered Streaming)
5	Non-GBR	1	100 ms	10^{-6}	IMS Signalling
6		6	300 ms	10^{-6}	Video (Buffered Streaming), TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)
7		7	100 ms	10^{-3}	Voice, Video (Live Streaming), Interactive Gaming
8		8	300 ms	10^{-6}	Video (Buffered Streaming), TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)
9		9			

2.2 QOS Mechanisms of WLAN

The original WLAN MAC layer doesn't support QoS, therefore, the IEEE proposed 802.11e protocol [1], which provides two channel access mechanisms to support the applications with QoS requirements: Enhanced Distributed Channel Access (EDCA) and HCF Controlled Channel Access (HCCA). In general, HCCA mechanism is not widely used at present.

In EDCA mechanism, the traffic with high priority is easier to be sent out than traffic with low priority. The traffic priorities are Access Categories (ACs), which are defined in EDCA to support the delivery of traffic with 802.1D User Priorities (UPs). The mapping between EDCA AC and 802.1D UP is shown in Table 2.

Table 2. The mapping between EDCA AC and 802.1D UP

Priority	802.1D UP	802.1D Designation	EDCA AC	Designation
Highest	7	NC (Network Control)	AC_VO	Voice
	6	VO (Voice)		
	5	VI (Video)	AC_VI	Video
	4	CL (Controlled Load)		
	3	EE (Excellent Effort)	AC_BE	Best Effort
	0	BE (Best Effort)		
	2	N/A	AC_BK	Background
Lowest	1	BK (Background)		

3 The DSCP-Based Method of QoS Class Mapping

Differentiated Services Code Point (DSCP) is defined by IETF [12] [13], used in DiffServ. The DS field in the IP header contains a 6-bit DSCP value to classify the different traffic. Generally, most networks use the four kinds of Per Hop Behaviors (PHBs) which is determined by DSCP.

- The Default PHB is used for the typically best effort traffics.
- The Expedited Forwarding (EF) PHB is used for the traffics with low loss and delay.
- The Assured Forwarding (AF) PHB provides the traffics with guaranteed delivery.
- The Class Selector (CS) PHB is used to be compatible with the IP Precedence field in the TOS byte.

Compared with IntServ, DiffServ has the advantages of scalability and simpleness, and it is discussed more in the heterogeneous networks at present.

In order to simplify the complexity of end-to-end QoS across heterogeneous networks, the GSM Association recommended using the DSCP, and specified the mapping between EPS QCI and DSCP [14], shown in Table 3.

Table 3. EPS QoS information and the mapping to DSCP values

EPS	QoS Information			IP Transport	
QCI	Traffic Class	THP	Signalling Indication	Diffserf PHB	DSCP
1	Conversational	N/A	N/A	EF	101110
2					
3					
4	Streaming	N/A	N/A	AF41	100010
5	Interactive	1	Yes	AF31	011010
6			No	AF32	011100
7		2	No	AF21	010010
8		3	No	AF11	001010
9	Background	N/A	N/A	BE	000000

Additionally, IEEE also considered the importance of QoS over WLAN interworked with the external networks, and proposed an example mapping of DSCP to EDCA ACs. The mapping table is shown in Table 4.

Table 4. Mapping of DSCP to 3GPP QoS information and EDCA ACs

3GPP QoS Information		DiffServ PHB	DSCP	QoS Requirement on GPRS Roaming Exchange				EDCA AC	UP
Traffic Class	THP			Max Delay	Max Jitter	MSDU Loss	MSDU Error Ratio		
Conversa-tional	N/A	EF	101110	20ms	5ms	0.5%	10^{-6}	AC_VO	7/6
Streaming	N/A	AF41	100010	40ms	5ms	0.5%	10^{-6}	AC_VI	5/4
Interactive	1	AF31	011010	250ms	N/A	0.1%	10^{-8}	AC_BE	3
	2	AF21	010010	300ms	N/A	0.1%	10^{-8}	AC_BE	3
	3	AF11	001010	350ms	N/A	0.1%	10^{-8}	AC_BE	0
Background	N/A	BE	000000	400ms	N/A	0.1%	10^{-8}	AC_BK	2/1

- QCI 1/2/3, used for conversational traffic, should be mapped to PHB EF (from Table 1 and Table 3), and should be mapped to EDCA AC_VO and UP 7/6 (from Table 4). However, according to Table 2, UP 7 is designated for NC (Network Control), so we suggest QCI 1/2/3 should be mapped to UP 6.
- QCI 4, used for streaming traffic, should be mapped to PHB AF41 (from Table 1 and Table 3), and should be mapped to EDCA AC_VI and UP 5/4 (from Table 4). Considering QCI 4/5/6 together, we suggest QCI 4 should be mapped to UP 5 to keep the priority order of UP shown in Table 2.
- QCI 5, used for signalling, should be mapped to PHB AF31 (from Table 1 and Table 3), and should be mapped to EDCA AC_BE and UP 3 (from Table 4). However, signalling has the highest priority in Non-GBR traffic, so QCI 5 should be mapped to EDCA AC_VI and UP 4.
- QCI 6, used for interactive traffic, should be mapped to PHB AF32 (from Table 1 and Table 3). However, QCI 6 is not included in Table 4. Considering QCI 5/6/7

together, we suggest QCI 6 should be mapped to EDCA AC_VI and UP4 to keep the priority order of UP.

- QCI 7, used for interactive traffic, should be mapped to PHB AF21 (from Table 1 and Table 3), and should be mapped to EDCA AC_BE and UP 3 (from Table 4).
- QCI 8, used for interactive traffic, should be mapped to PHB AF11 (from Table 1 and Table 3), and should be mapped to EDCA AC_BE and UP 0 (from Table 4).
- QCI 9, used for background traffic, should be mapped to PHB BE (from Table 1 and Table 3), and should be mapped to EDCA AC_BK and UP 2/1 (from Table 4). However, according to Table 2, UP 2 is not designated, so we suggest QCI 9 should be mapped to UP 1.

Considering the above, we present the DSCP-based method of QoS class mapping between WLAN and EPS network, listed in Table 5.

Table 5. QoS class mapping between EPS QCI, DSCP, 802.1D UP and EDCA AC

EPS QCI	Resource Type	Diffserv PHB	DSCP	802.1D UP	EDCA AC	Services
1	GBR	EF	101110	6	AC_VO	Conversational Voice
2	GBR	EF	101110	6	AC_VO	Conversational Video
3	GBR	EF	101110	6	AC_VO	Real Time Gaming
4	GBR	AF41	100010	5	AC_VI	Buffered Streaming
5	Non-GBR	AF31	011010	4	AC_VI	Signalling
6	Non-GBR	AF32	011100	4	AC_VI	Buffered Streaming
7	Non-GBR	AF21	010010	3	AC_BE	Interactive Gaming
8	Non-GBR	AF11	001010	0	AC_BE	WWW, etc.
9	Non-GBR	BE	000000	1	AC_BK	FTP, E-mail, etc.

Analyzing the QoS parameters (e.g., the IP packet delay, IP packet jitter and IP packet error/loss rate) of different QoS classes over the heterogeneous networks, we consider that the method of QoS class mapping is feasible and effective. Because the end-to-end QoS class mapping is concave (shown in equation (1)), generally its granularity is subject to the network with coarse-grained QoS. Therefore, the granularity of this method is depended on WLAN. The simulations and experiments are discussed in the following section.

$$c(u, v) = \min\{c(u, u_1), c(u_1, u_2), \cdots, c(u_k, v)\} \tag{1}$$

4 Simulation and Analysis of QoS Class Mapping

4.1 Simulation Scenarios

The OPNET is an excellent simulation platform, which is used to analyze and design communication networks, devices, protocols, and applications. However, the current version of OPNET does not support PCC architecture, so we design and implement the PCC in OPNET. In this paper, in order to analyze the feasibility of the DSCP-based method of QoS class mapping, we design the simulation scenarios, and simulate them in OPNET. The simulation of network architecture is shown in Fig. 2.

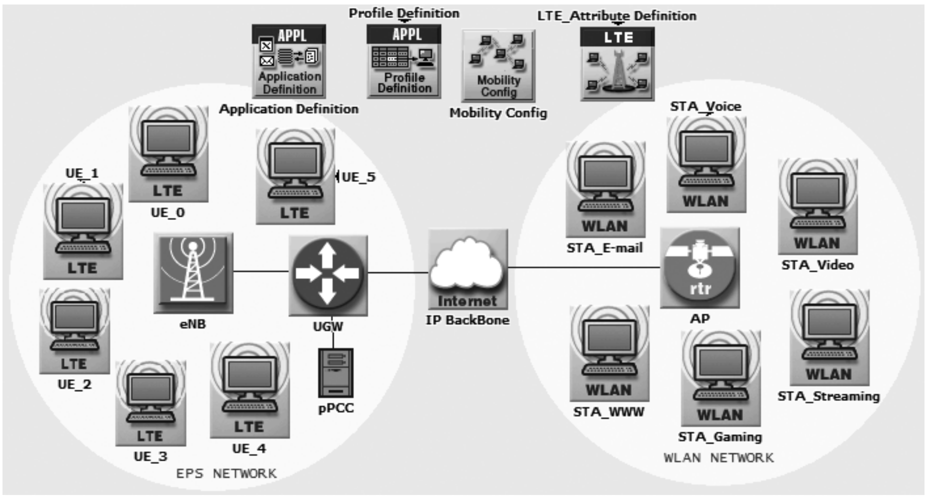


Fig. 2. Simulation of network architecture in OPNET

The simulated EPS network is composed of UEs, eNodeB, UGW and pPCC. The UGW is the unified packet gateway, which provides the combined functions of Serving-GW and PND-GW. The UGW is interworked with the IP backbone of Internet. The pPCC is the subset function of PCC, which is focused on QoS to provision the policy control.

The simulated WLAN network is composed of STAs and AP, which all support the EDCA mechanism. The AP is interworked with the IP backbone of Internet. The STAs communicate with the UEs in EPS networks.

According to Table 5, we design 6 typical services: conversational voice, conversational video, streaming, interactive gaming, WWW and Email. The 6 services are running simultaneously during simulating. The UGW do the mapping function between EPS QCI and DSCP, and the AP do the mapping function between DSCP and EDCA AC.

4.2 Experimental Results

Every kind of traffic has its own applicable QoS requirements. ITU-T Y.1541 [15] recommended IPTD (IP Packet Transfer Delay), IPDV (IP Packet Delay Variation), IPLR (IP Packet Loss Ratio) and IPER (IP Packet Error Ratio) as QoS evaluation metrics.

In this paper, we choose packet one-way-delay, packet delay variation as key parameters of QoS requirements. Services of streaming, video and voice are sensitive to packet one-way-delay and packet delay variation, however, services of Email, WWW and interactive gaming are not concerned with packet delay variation.

The simulation result of packet end-to-end delay of streaming, video and voice is shown in Fig. 3. From the result, the delay of streaming is higher than video, and the delay of video is higher than voice. The delay without mapping is higher than the delay with mapping.

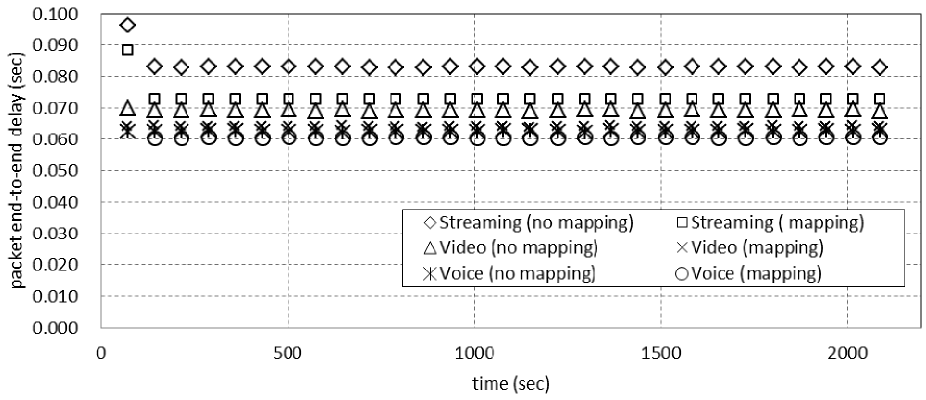


Fig. 3. Packet end-to-end delay of streaming, video and voice

The simulation result of packet end-to-end delay of Email, WWW and interactive gaming is shown in Fig. 4. From the result, the packet end-to-end delay of Email is highest among these three services. Because Email is an uncontinuous background service, the delay of it without mapping is lower than the delay with mapping generally. The delay of WWW is higher than interactive gaming. The delays of WWW and interactive gaming without mapping are a little higher than the delays with mapping.

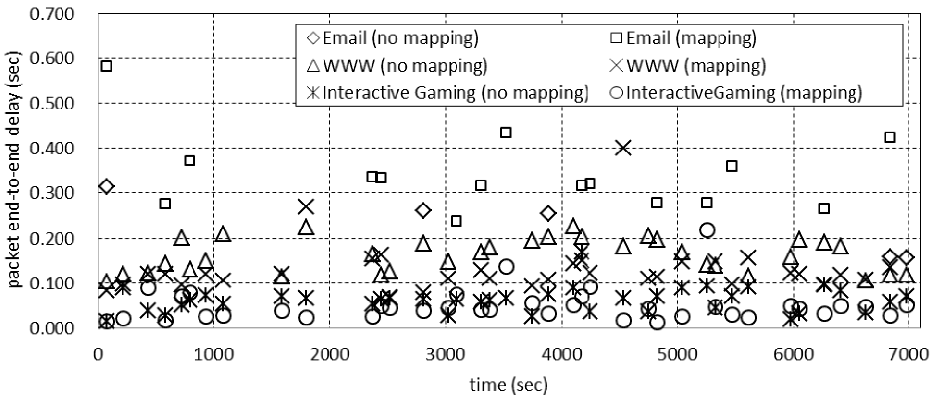


Fig. 4. Packet end-to-end delay of Email, WWW and interactive gaming

The simulation result of packet delay variation of streaming, video and voice is shown in Fig. 5. From the result, the packet delay variation of streaming is highest among these three services, and the delay variation of it without mapping is higher than the delay variation with mapping. The delay variations of video and voice without mapping are a little higher than the delay variations with mapping, which all are very close to zero.

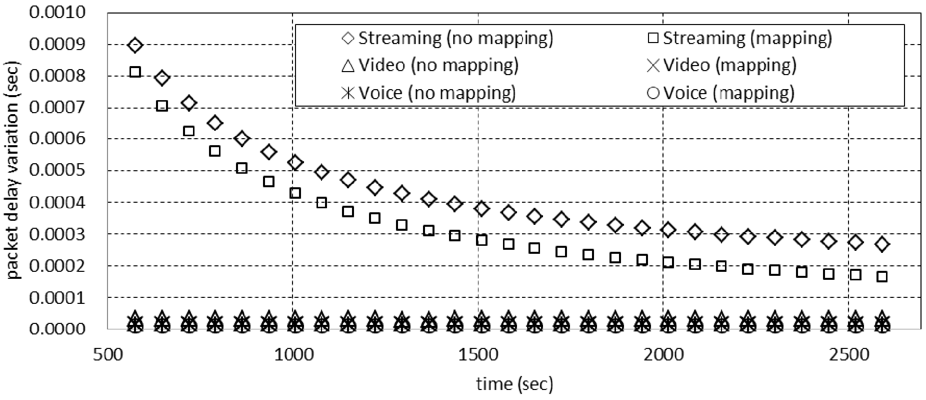


Fig. 5. Packet delay variation of streaming, video and voice

According to the experimental results shown above, the DSCP-based method of QoS class mapping is conformed to the expectation generally.

5 Conclusions

In this paper, we investigate the issue on how to provide the end-to-end QoS guarantees between WLAN and EPS network, which both are interworked with Internet. We study the current researches, and analyze the QoS mechanisms of LTE and WLAN. Then we propose the DSCP-based method of QoS class mapping between WLAN and EPS network. By simulating and analyzing the QoS class mapping method in OPNET, we consider that the method is conformed to the expectation.

In future, we will investigate the end-to-end QoS guarantees over more comprehensive heterogeneous networks. Meanwhile, we will design more types of services in OPNET and take more QoS metrics (e.g., packet loss ratio and packet error ratio) into account.

Acknowledgments. The research is supported by Science and Technology Breakthrough Plan of Shanghai Science and Technology Committee (No. 11dz1500205) and Scientific Research and Innovation Fund Project of East China Normal University (No. 78210253).

References

1. 802.11 standard, http://standards.ieee.org/about/get/802/802.11.html
2. 3GPP: Policy and charging control architecture; V11.7.0 Release 11. TS 23.203, 3rd Generation Partnership Project (3GPP) (2012)
3. Ryu, M., Kim, Y., Park, H.-S.: Systematic QoS Class Mapping Framework over Multiple Heterogeneous Networks. In: Balandin, S., Moltchanov, D., Koucheryavy, Y. (eds.) NEW2AN 2008. LNCS, vol. 5174, pp. 212–221. Springer, Heidelberg (2008)
4. Wang, Z.J., Dong, Y.N., Shi, H.X., Zhang, H.: Modeling and Analysis of QoS Class Mapping for Hybrid QoS Domains Using Flow Aggregate. In: 9th International Wireless Communications and Mobile Computing Conference, pp. 503–508. IEEE Press, New York (2013)
5. Oh, S.-M., Kim, J.-H., Hwang, Y.-S., Kwon, H.-Y., Park, A.-S.: End-to-End QoS Guaranteed Service in WLAN and 3GPP Interworking Network. In: Kim, Y.-T., Takano, M. (eds.) APNOMS 2006. LNCS, vol. 4238, pp. 180–189. Springer, Heidelberg (2006)
6. Sarraf, C.M., Ousta, F., Yusoff, M.Z., Kamel, N.: Mapping Quality of Service Classes between UMTS, WiMAX and DiffServ/MPLS Networks. In: 2013 World Congress on Computer and Information Technology, pp. 1–5. IEEE Press, New York (2013)
7. Horvath, G., Fazekas, P.: End-to-end QoS signaling for LTE. In: 2013 21st Telecommunications forum, pp. 82–85. IEEE Press, New York (2013)
8. Chen, C.L.: A Proposal of Next Generation Network: QoS Mapping for MPLS-DiffServ and Label Forwarding. In: 2012 5th International Conference on BioMedical Engineering and Informatics, pp. 1416–1419. IEEE Press, New York (2012)
9. Muntean, V.H., Otesteanu, M., Muntean, G.M.: QoS Parameters Mapping for the E-learning Traffic Mix in LTE Networks. In: 2010 International Joint Conference on Computational Cybernetics and Technical Informatics, pp. 299–304. IEEE Press, New York (2010)
10. Liu, Y., Lu, G., Zhang, W., Cai, F.L.: The QoS class mapping in WLAN and 3GPP LTE Interworking Network. In: 2013 3rd International Conference on Computer Science and Network Technology, pp. 898–901. IEEE Press, New York (2013)
11. Olsson, M., Sultana, S., Rommer, S., Frid, L., Mulligan, C.: EPC and 4G Packet Networks: Driving the mobile broadband revolution, 2nd edn. Academic Press, Waltham (2013)
12. Nichols, K., Blake, S., Baker, F., Black, D.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (1998)
13. Davie, B., Charny, A., Bennett, J.C.R., Benson, K., Le Boudec, J.Y., Courtney, W., Davari, S., Firoiu, V., Stiliadis, D.: An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246 (2002)
14. GSM Association Official Document IR.34v9.1: Guidelines for IPX Provider networks (Previously Inter-Service Provider IP Backbone Guidelines) (2013)
15. ITU-T Recommendation Y.1541: Network performance objectives for IP-based services (2011)

HostoSink: A Collaborative Scheduling in Heterogeneous Environment

Xiaofei Liao, Xiaobao Xiang, Hai Jin, Wei Zhang, and Feng Lu

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

Abstract. Due to the limitations of power consumption and memory capacity, the past few years have observed a strong trend of using heterogeneous environment equipped with accelerators, such as GPU (Graphic Processing Unit) and FPGA (Field Programmable Gate Array), and even MIC (Many Integrated Core), to help the traditional SMP (Symmetric Multi-Processing) CPU to speed up applications. In this paper, we choose the Intel MIC architecture coprocessor as the accelerator and design HostoSink, a runtime system for collaborative scheduling based on Pthread task. With the help of runtime characteristics of the application and the heterogeneous environment for scheduling the Pthread tasks between CPU and MIC automatically and dynamically, HostoSink provides MIC users with an easier way to gain high performance in heterogeneous CPU-MIC environment without the need of optimizing the original Pthread-based multi-threaded applications manually too much. Experimental results show that by using HostoSink, the overall speedup can achieve more than 3x speedup compared with the original performance by using CPU only and the average amount of data transmission between CPU and MIC is also reduced.

Keywords: Heterogeneous Environment, Runtime System, Task Scheduling, MIC.

1 Introduction

Due to the limitations of some physical factors, such as power, interconnection delay and chip complexity, the scalability of traditional SMP CPU architecture faces great challenges. This situation makes the usage of variety of accelerators more and more popular. GPU implements new programming models and languages for high performance computing, such as CUDA (Compute Unified Device Architecture) mainly developed for the NVIDIA series [1] and OpenCL (Open Computing Language) charged by the Khronos Group [2]. Some examples of these works are Quantum Monte Carlo, Artificial Intelligence [3], and Ray Casting Simulation on GPU [4]. FPGA is another choice due to its flexibility and high performance over the years, even though it is not such a common hardware as GPU. FPGA itself is semi-programmable because it is composed of many logic blocks, which are configurable and interconnected by a reconfigurable general routing structure [5]. Although these accelerators are widely used, there are still some obstacles that hinder the efficient application design and development in heterogeneous environment equipped with such accelerators.

Traditional accelerators and SMP CPU are built based on very different philosophies. To pack a huge number of cores (or processing units), traditional accelerators abandon some advanced features, such as branch prediction, out-of-order execution and super scalar, to make cores (or processing units) much simpler [6]. Therefore, they are completely different in architecture, instrument type (or flow) and hardware, which do not allow existing executable binaries run directly on them. To generate correct executable binaries, application designers should learn one or more programming models, languages and some other special skills as it may be even devoted for very experienced users. Moreover, as most of traditional accelerators are just designed for some specific areas, the limited range of their usage is also a big problem. In addition, how to make full use of the heterogeneous system's capacity is another challenge. First, traditional accelerators are not designed for general purpose computing, and only parts of the application are suitable for running on the accelerators. In order to guarantee the high performance, we should have a glimpse of both the application' and the accelerators' features before redesign them for the heterogeneous environment. Second, to utilize the potentialities of the heterogeneous environment maximally, it is very important to schedule the suitable parts properly to the accelerators, but the traditional scheduling strategies are always static solutions which mean that they determine the scheduling during design time and generate the same scheduling result at run time when run the application every time. Though these strategies can remove some transient task switching and data transmission overhead, the scalability and flexibility are greatly affected.

To overcome the aforementioned obstacles, there is a growing demand of exploring new heterogeneous environment equipped with new accelerator and corresponding programming models and languages to make the application design and development easier and more efficient. Our approach is to use the Intel SMP CPU (Intel Xeon Product Family) and the Intel MIC architecture coprocessor (Intel Xeon Phi Product Family) as the accelerator to construct the heterogeneous hardware environment. For convenience, we also call SMP CPU as host-side and accelerator as sink-side in our system. Furthermore, based on this environment, we design and implement an absolutely new runtime system, HostoSink, which can schedule the Pthread task between host-side and sink-side at runtime automatically and dynamically under the premise of ensuring high speedup and high resources utilization rate of the overall system. On the whole, there are three basic principles to achieve the targets above.

1) Statically analyzing and marking the source code of the application to tag the potential tasks and initialize the scheduling strategy. Then, generating and managing the executable binaries of each potential task both for host-side and sink-side.

2) Carefully handling the data transmission between host-side and sink-side in order to reduce the overall overhead of the system.

3) Dynamically and automatically scheduling the tasks between host-side and sink-side according to the runtime characteristics of the application and the heterogeneous environment.

With some preliminary configuration, HostoSink does not require application designer to modify the original Pthread-based multi-threaded applications codes too much in order to run the application on this heterogeneous CPU-MIC environment efficiently. Meanwhile, through using the improved data transmission strategy, the amount of data transmission between host-side and sink-side is also reduced.

The rest of this paper is organized as follows. Section 2 discusses the related works. The detail designation and design principles are described in Section 3. Section 4 describes the construction of our original system HostoSink and presents the performance evaluation. Then, we conclude this work in Section 5.

2 Related Works

In this section, we give a brief overview of the related technologies including the development of accelerators, typical programming models and languages, and different runtime systems in heterogeneous environment.

The Top500 list in November 2013 [7], as well as recent announcements by the TACC (Texas Advanced Computing Center) and ORNL (Oak Ridge National Laboratory) [8], shows that the next generation supercomputers will use accelerators to speed up computing instead of using traditional SMP CPU only. Besides the typical accelerators mentioned above, there are still many other novelty attempts. Fan [9] has proposed a kind of accelerator for multiple similar loops through high level of customization and semi-programming to achieve power efficiency and computing speedup. Platune [10] is an open source and parametric multicore accelerator, which allows designers to parametrically configure it to achieve the goals of performance and power, etc. Furthermore, simulation results of the application are also available. Tan [11] leads the research and development of a heterogeneous platform RAMP (Research Accelerator for Multiple Processors) using FPGA, which is used to study multicore accelerators (more than 16 physical cores) based on FPGA.

In early 2013, Intel releases its commercial MIC architecture product, codenamed Knights Corner, the branding of the processor product family as Intel Xeon Phi [12]. It is worth mentioned that there are 13 of 53 heterogeneous supercomputers in the Top500 list, which use MIC architecture including China's Tianhe-2 Supercomputer until November 2013 [7]. Built on simple x86 cores and some customized components, the design of MIC allows using standard and existing programming models and languages.

At the beginning, the heterogeneous environment refers to CPU-GPU environment mainly and uses Brook or Cg as its programming language [13]. Then, NVIDIA promotes CUDA for its GPU products in order to support them much better. CUDA provides a software environment in which developers can use C language as the main programming language. Moreover, its programming model supports not only automatically extending the C language style source code to heterogeneous CPU-GPU environment, but also transparently scaling the existing CUDA application with the increasing number of processing units provided by GPU. Just not too long after CUDA, Apple develops OpenCL initially, and refines it in collaboration with many technical teams from AMD, IBM, Intel and NVIDIA, and then submits it to the Khronos Group as an open and royalty-free standard. In July 2013, Khronos Group released OpenCL 2.0 specification, which shows the standard is becoming more and more mature and acceptable. Essentially, OpenCL compiles and manages kernels, which are actually functions to be executed on GPU at runtime to enable the application to make full use of the computing resources. With Intel's promotion of MIC architecture, they provide the corresponding programming models and languages for heterogeneous CPU-MIC environment simultaneously. Using extended C, C++ and FORTRAN language, we can use the environment in five forms from CPU-centric style to MIC-centric style including multicore hosted style, offload style, symmetric style, reverse offload style (not supported yet) and MIC hosted style according to the applications' characteristics (serial, highly parallelized, or partially parallelized). Based on IA (Intel Architecture), the MIC architecture supports all programming models for traditional IA CPU, such as Pthread (POSIX thread), OpenMP (Open Multi-Processing), MPI (Message Passing Interface) and TBB (Intel Threading Building Blocks) [14]. In spite of the differences between instruction sets, we can efficiently generate MIC executable binaries by #pragmas primitive, array syntax (used for vectorization) or intrinsic functions manually.

In addition, many other commercial solutions, such as MDC (Microsoft's DirectCompute) and Intel's ArBB (Array Building Blocks) and many valuable research attempts, such as DSL (Domain Specific Languages) [15] and explorative programming model [16] are also available.

Essentially, the runtime systems are mainly responsible for deciding how and when to generate, manage and schedule the tasks.

From the perspective of function design, there are a variety of runtime systems. Farooqui [17] presents a real-time binary translation system for heterogeneous CPU-GPU environment, which is not suitable for newly proposed platforms. Sujeeth [18] presents OptiML, which parallels application implicitly and automatically through machine learning. However, it is only suitable for specific DSL as they provide. Gelado [19] designs a runtime system for memory sharing between CPU and GPU which allow CPU to read GPU's memory directly, not vice versa, which shows that it is still an exploratory work. There are some other similar works, which also design the runtime system from memory management area [20][21]. Besides, Qin [22] presents a dynamic virtual execution system for executing existing general-purpose binaries without any modification and recompilation over heterogeneous CPU-GPU environment, but they just care the functionality, not the overall performance.

From the perspective of performance goals, many task-scheduling strategies in the heterogeneous runtime systems are studied. Augonnet [23] presents a unified platform, StartPU, used for unusual heterogeneous multicore environment. Winter [24] schedules thread for heterogeneous many-core environment to achieve power efficiency, but the system's overall performance is not their emphasis. Their sample-run model brings too much overhead. Besides, to make full use of the environment's resources, Song [25] focus on the load balance between the heterogeneous components. Similarly, Bartzas [26] presents their own data management strategy in heterogeneous multicore environment to improve their system's performance.

3 Design of HostoSink

3.1 System Overview

As shown in Figure 1, the system is designed for large workloads and each of them can be split into one or more tasks. Before an application runs in the environment for the first time, the original source code of the application is analyzed briefly and automatically to generate the potential tasks and the initial strategy for the first scheduling. Respectively, we need to record the basic task information, such as total task number, start time and characteristics of each task, in each step for the later use. At runtime, we monitor and analyze runtime characteristics for both the application and the heterogeneous environment at first, and then record them similarly time-to-time. According to the information recorded by the profiler, we regenerate and reschedule each task to ensure high speedup and high resource usage of the whole system. Meanwhile, we also use some improved ways to reduce the amount of data transmission between host-side and sink-side.

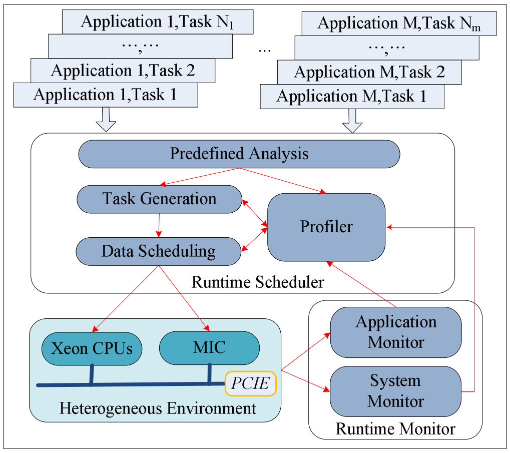


Fig. 1. System architecture overview

Overall, the entire system is a closed loop system, which can maximize its performance as much as possible through self-regulation and self-adjustment.

3.2 Task Generation and Management

Task Granularity. The first factor we have to decide is what kind of task granularity we should choose for our task generation later. Generally speaking, we can classify the granularity into application level, process level, thread level and instruction level from top to down. In our system, we choose the thread level granularity for three reasons:

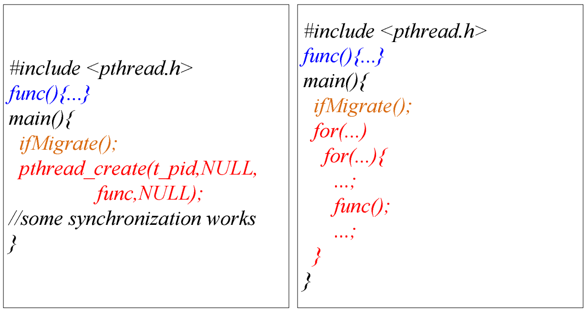
1) Application level or process level granularity is too coarse for task scheduling, therefore the flexibility and accuracy of the system is undetermined. On the contrary, instruction level granularity is fine-grained, but it is not practical since we must conduct real time binary translation to bridge the gap between the instruction sets.

2) Thread level parallelization is much more common as it not only demands much less resources than application or process level parallelization but also is more operational than instruction level parallelization which meets our original goal of efficient scheduling.

3) Thread is the basic scheduling unit of the operating system and the basic functional unit of the processor. So, multi-threaded application is much easier to be scaled with more computing resources.

As we have chosen thread level granularity, the next step is to decide which kind of thread we should choose in our system. Pthread and OpenMP are the two typical libraries that support multi-threaded programming. However, the scheduling scheme of threads is controlled by runtime system when using OpenMP library. So, we choose the Pthread-based thread to generate our target tasks.

Task Generation: As Intel has stated, MIC is mainly designed for scientific application of highly degreed parallelism. Correspondingly, we should extract the suitable parts of the application and run them on sink-side. In our system, we mainly focus on two types of typical parallel-style codes, which are used to generate our Pthread task. One is the typical Pthread style code as shown in Figure 2(a). The other is the common for-loop style code as shown in Figure 2(b).



```
#include <pthread.h>
func(){...}
main(){
  ifMigrate();
  pthread_create(t_pid,NULL,
               func,NULL);
//some synchronization works
}
```

```
#include <pthread.h>
func(){...}
main(){
  ifMigrate();
  for(...)
    for(...){
      ...;
      func();
      ...;
  }
}
```

(a) (b)

Fig. 2. Typical code for generating tasks

For codes like Figure 2(a), the system just simply extracts the *pthread_create* function and other codes it depends on, packs them into a separate sourcefile. To support runtime scheduling, the function *ifMigrate* is inserted before *pthread_create* function to make the decision. At compiling time, the system compiles these source codes into two binary versions, one for host-side and the other for sink-side. For codes like Figure 2(b), if the inner loop can be vectorized well and its iteration time is large enough, the system will divide it into many parallel for-loops and pack them into separate threads to generate similar style codes as Figure 2(a). Then, the system will deal with it in the same way as Figure 2(a).

Task Management: As mentioned above, our system is designed for large scale of workloads, so a large number of tasks exist in the system simultaneously. To manage these tasks correctly and orderly, we use a tuple *<ID, TYPE, BF, TP, ST, LRT1, LRT2, CT, STATE>* to represent a task in the system. Detailed meaning of each element in the tuple is as follows:

- *ID*: Unique identifier of a task;
- *TYPE*: Type of the task (Pthread or for-loop);
- *BF*: Binary file path of the task;
- *TP*: Target side to run the task;
- *ST*: Start time of the task;
- *LRT1*: Last run time on host-side;
- *LRT2*: Last run time on sink-side;
- *CT*: CPU time on host-side;
- *STATE*: State of the task.

3.3 Data Transmission and Management

It is known that, the application is a combination of code and data. In previous section, we have proposed our task generation and management methods, which solve the problem of code management. Correspondingly, we also need to solve the problem of data management. That is, before we start the task, we have to place the data the task needed in the right place, and after the task has done, we have to write back the result to the right place, too. Here, the data refers to memory data in our system.

Using the x16 Gen2 PCIE interface, Intel claims that the data transmission capability between CPU and MIC can reach as fast as 6.7GB/s from CPU to MIC and 6.9GB/s from the contrary direction. Nevertheless, we still need to manage the data operation carefully to improve the data transmission efficiency and avoid the unexpected mistakes caused by data synchronization as far as possible especially when the data transmission is frequent and dramatic. In our system, we use a tuple *<ID, T_ID, FROM, TO, SIZE>* to represent a data transmission. Detailed meaning of each element in the tuple is as follows:

- *ID*: Unique identifier of a data transmission;
- *T_ID*: *ID* of the corresponding task;
- *FROM*: Source of the data transmission;
- *TO*: Destination of the data transmission;
- *SIZE*: Size of the data to be transmitted.

For simplicity, our system treats host-side as the master device and sink-side as the slave device which means that data is stored in host-side originally and can be transmitted from host-side to sink-side before the task starts and is written back to host-side after the task finishes. This simplification makes data management more intuitive and easier. When a task is scheduled to sink-side, the typical data transmission situations are shown in Figure 3.

Figure 3(a) represents the most common and easiest situation. *Task i* is scheduled to MIC and read *Data m* from host-side, then write data back to host-side as *Data n*. *Task j* acts in the same manner and is independent of *Task i*. For this case, we just simply transmit data of each task at the right time. Moreover, if the amount of data transmission is small and the two tasks are close in time, we can do the transmission together to reduce communication overhead. Besides, to avoid the situation that a task is blocked for data transmission, we can appropriately transmit data in advance according to the task's *ST* attribute. Figure 3(b) represents a situation in which *Data n* is not only the output of *Task i*, but also the input of *Task j*. To reduce data transmission time, we do not

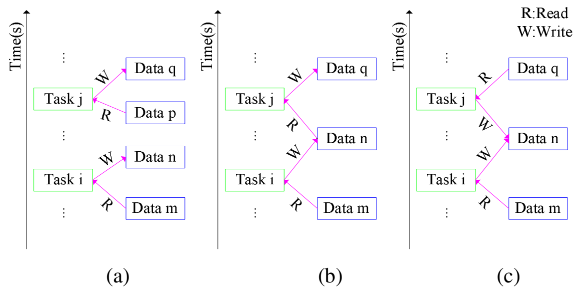


Fig. 3. Data transmission situations

write back *Data n* actually if *Data n* is just an intermediate result. For example, when an array is partitioned for several tasks, we just need to transfer the whole array to sink-side for the first task and write back the result for the last one. Figure 3(c) shows a typical situation of data conflict, we do not do any optimization for this situation.

3.4 Runtime Task Scheduling

Generally speaking, the main purpose of task scheduling is to decide when and where (host-side or sink-side) to run the task. Specifically, task scheduling is responsible for considering all conditions used to comprehensively make scheduling decision. After task scheduling, our system loads and starts the task at the right time and the right side. Similar to the problem of data transmission, we also treat host-side as the master device and sink-side as the slave device. So, the tasks tend to run in host-side originally and be scheduled to sink-side if needed.

Motivation: As shown in Figure 4, the main advantage of MIC is its high thread scalability. As it can provide up to 60 physical cores and 240 logical threads, application, which can be extended to more than 120 threads, can achieve significant performance improvement. Besides, the VPU (Vector Processing Unit) of each core is extended to 512 bits wide, which is much wider than Intel's previous products as shown in Figure 5. However, simple physical core means low processor frequency and poor single-threaded capacity. Each core's frequency of MIC is as low as 1.05GHz. In view of these, collecting characteristics of the application and scheduling

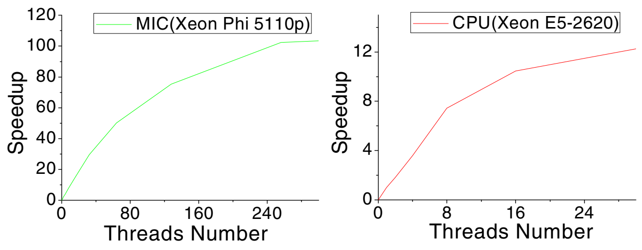


Fig. 4. Thread scalability of MIC and CPU

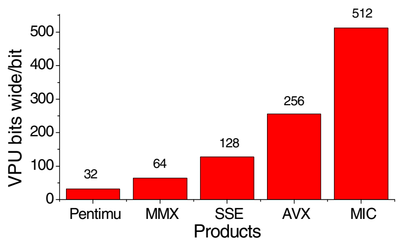


Fig. 5. VPU bits wide comparation between Intel products

the suitable tasks the application generating at sink-side and leaving the unsuitable ones in host-side at runtime can provide considerable performance. Moreover, we also use the heterogeneous environment's runtime characteristics as an auxiliary scheduling condition to ensure an acceptable load difference between host-side and sink-side.

Task Scheduling Strategy: Before we introduce our task scheduling strategy, we need to do some brief preparations.

First, to facilitate task scheduling, we organize all the tasks in a FIFO queue. Two operations can be performed on that queue: task acceptation (push) and task submission (pop). All the tasks are generated at host-side and pushed to the queue. Then, we schedule the tasks at the beginning of the queue before it is popped out and started. The whole procedure of task scheduling is shown in Figure 6.

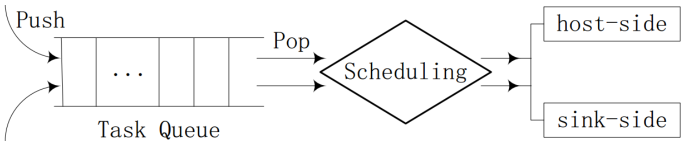


Fig. 6. Main procedure of task scheduling

Second, we abstract the structure of the application for simplification as shown in Figure 7. In order to collect runtime characteristics of the application, the application should own multiple kinds of thread and each kind should run multiple times. Besides, each task contains one or more threads with the same style.

```
#include <...>
func1(){...}
func2(){...}
...
funcn(){...}
main(){
 for(...){
  for(...){
    pthread_create(t_pid1,NULL,
           func1,NULL);
 }
  for(...){
    pthread_create(t_pid2,NULL,
           func2,NULL);
 }
  ...;
  for(...){
    pthread_create(t_pidn,NULL,
           funcn,NULL);
 }
 }
}
```

Fig. 7. Simplified structure of application

Third, to describe our task scheduling strategy better, we make some conventions as follows:

Definition 1: Computation Complexity (*CC*). We calculate *CC* as (1). We also assume that the lower acceptable limit of CC for sink-side is *minCC*.

$$CC = \frac{CT}{LRT1} \tag{1}$$

Definition 2: Vectorization Degree (*VD*). We assume that the *ith* vectorized code segment's predicted speedup of a task is S_i, the elapsed time is T_i and the task contains *n* similar segments. We calculate *VD* as (2). We also assume that the lower acceptable limit of *VD* for sink-side is *minVD*.

$$VD = \frac{\sum_{i=1}^{n} S_i * T_i}{\sum_{i=1}^{n} T_i} \tag{2}$$

Definition 3: Workload of Host-side (*WoH*). We assume that utilization rate of host-side cores is *Uc* and the memory utilization rate is *Um*. We calculate *WoH* as (3). We also assume the upper acceptable limit of load difference is *maxW*.

$$WoH = \frac{1}{(1-U_c)*(1-U_m)} \tag{3}$$

Definition 4: Workload of Sink-side (*WoS*). We calculate *WoS* in the same way as *WoH*.

We describe our scheduling strategy as shown in Figure 8. After the default configuration for task scheduling, we pop a task from the queue. Then we compare the *CT* attribute of the task with zero to determine if we run the task for the first time (*CT* > 0) or not. If yes, we just schedule it to host-side to collect its runtime characteristics; else we extract its runtime characteristics that have been recorded for subsequent scheduling steps. Specifically, we schedule the task whose *CC* attribute is less than *minCC* and *VD* attribute is less than *minCD* to host-side. Similarly, we also schedule the task whose CC attribute is greater than *minCC* and *VD* attribute is greater than *minVD* to sink-side. Then, we schedule the other types of task through the magnitude relationship between the current *WoH* and *WoS* of the heterogeneous environment. If the difference between *WoH* and *WoS* is greater than *maxW*, we schedule the task to sink-side. Otherwise, we schedule it to host-side. After the task finishes, if we receive the termination signal to stop the scheduling service, we should close the system; otherwise we pop a new task and schedule it as above.

Experimental results show that the scheduling strategy can work correctly and efficiently in heterogeneous CPU-MIC environment.

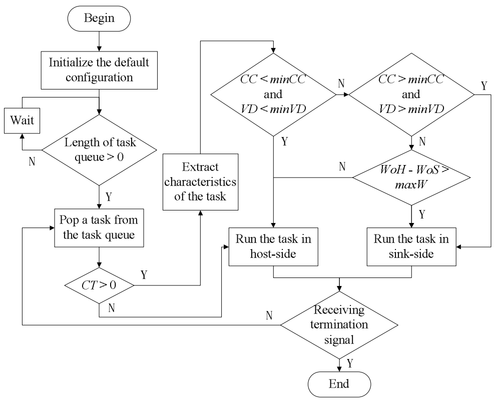

Begin
Initialize the default configuration
Wait
Length of task queue > 0
N
Y
Pop a task from the task queue
CT > 0
Y
N
Extract characteristics of the task
CC < minCC and VD < minVD
N
Y
CC > minCC and VD > minVD
Y
N
WoH - WoS > maxW
N
Y
Run the task in host-side
Run the task in sink-side
Receiving termination signal
N
Y
End

Fig. 8. Runtime task scheduling strategy

4 Evaluation and Experimental Results

4.1 Experimental Environment

The experiments are run in a real heterogeneous CPU-MIC environment and the hardware configurations and software requirements are shown in Table 1. We use Intel Xeon E5-2620 CPU as our host-side processor and Intel Xeon Phi 5110p as our sink-side coprocessor. Besides, we use Intel C++ compiler to generate MIC version binaries and install Intel MPSS (Many-core Platform Software Stack) to support underlying interaction.

Table 1. Parameters of experimental setup

Host-side	Hardware	Intel Xeon E5-2620, Processor: 12 cores, 24 threads, 2.10GHz, 256 bits VPU, Memory: 128GB (DDR3)
	OS	Redhat Enterprise Linux 6.0 (2.6.32-x86_64)
	Software	Compiler: Intel icpc (14.0.0), MPSS: mpss-3.1.1-rhel-6.0
Sink-side	Hardware	Intel Xeon Phi 5110p, Processor: 60 cores, 240 threads, 1.05GHz, 512 bits VPU, Memory: 8GB (DDR5)
	OS	Intel µOS (Linux)
Benchmark	Customized applications with hybrid thread types of different computation complexity and vectorization degree	

4.2 Speedup

As Intel has declared, the heterogeneous CPU-MIC environment can provide up to 3x greater performance on some highly parallel applications through manual optimization compared with Intel Xeon E5 series servers. The experiments show that our system can achieve approximate performance without manual optimization. Specifically, our benchmark contains three kinds of thread. Thread1 is suitable for MIC. Thread2 is suitable for CPU. Thread3 works as the background workload and is scheduled according to the load difference.

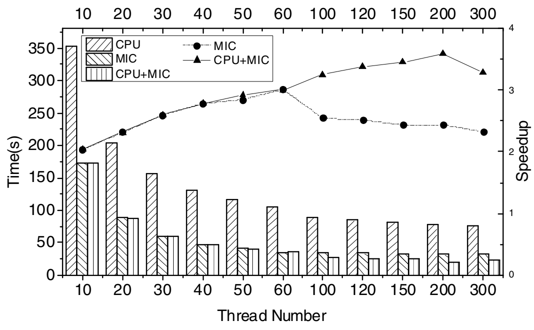


Fig. 9. The impact on performance of the number of thread1

Figure 9 shows the relationship between the number of thread1 and the corresponding performance improvement. We can see that, with the growing number of thread1, the heterogeneous CPU-MIC environment can achieve more than 3x greater speedup compared with using CPU only. Although the performance declines slightly when there are too many threads, the scalability of heterogeneous CPU-MIC environment is still better than using MIC only because of CPU's strong control capacity.

Figure 10 shows the relationship between the number of thread2 and the corresponding performance improvement. We can see that, with the growing number of thread2, the heterogeneous CPU-MIC environment can achieve more than 60x greater speedup compared with using MIC only although the performance declines. We can also find that the performance of the heterogeneous CPU-MIC environment is still much better than just using CPU only because MIC can also take over parts of tasks belong to CPU original.

In short, the experiments prove not only the efficiency of our system but also the importance of scheduling the tasks to where they are suitable.

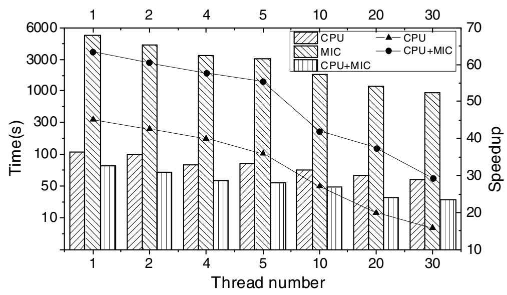


Fig. 10. The impact on performance of the number of thread2

4.3 Data Transmission

To prove the efficiency of our improved data transmission strategy, we conduct a 1024x1024 matrix multi-threaded multiplication test (element of the matrix is double precision floating point number). As shown in Figure 11, our improved data transmission strategy can reduce a large amount of data transmission compared with the original strategy.

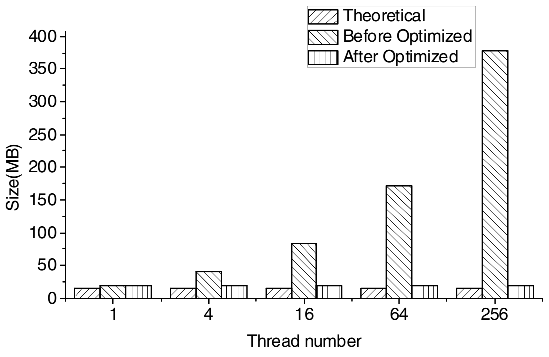


Fig. 11. Test of the efficiency of our improved data transmission strategy

5 Conclusions

This paper aims to provide a collaborative scheduling system, HostoSink, based on Pthread task in heterogeneous CPU-MIC environment. We schedule the Pthread tasks between CPU and MIC automatically and dynamically by using the runtime characteristics of the application and the heterogeneous environment. Our work is implemented in a real heterogeneous CPU-MIC environment and the experimental results show that by using HostoSink, the overall speedup can achieve more than 3x greater compared with the original performance by using CPU only and the average amount of data transmission between CPU and MIC is also reduced through using the improved data transmission strategy of our system.

In the future, we will optimize our system in following fields: first, we will extend our system to support more than one MIC coprocessors to make it more practical. Second, we will take more runtime characteristics of application such as cache miss ratio into consideration for task scheduling. Third, the future version will support task live migration improve the flexibility of our system.

Acknowledgment. This paper is supported by China National Natural Science Foundation under grant Nos.61272408, 61322210, National High-tech Research and Development Program of China (863 Program) under grant No.2012AA010905, Doctoral Fund of Ministry of Education of China under grant No.20130142110048.

References

[1] CUDA documents, `http://developer.download.nvidia.com/compute/cuda/docs/CUDA_Architecture_Overview.pdf`

[2] John, E.S., David, G., Shi, G.: OpenCL: A parallel programming standard for heterogeneous computing systems. IEEE Science & Engineering Magazine 12(3), 66–68 (2010)

[3] Scanniello, G., Ugo, E., Giuseppe, C., Carmine, G.: Using the GPU to Green an Intensive and Massive Computation System. In: 17th IEEE European Conference on Software Maintenance and Reengineering (CSMR), pp. 384–387. IEEE Press (2013)

[4] Xiao, S., Balaji, P., Dinan, J., Zhu, Q., Thakur, R., Coghlan, S., Lin, H., Wen, G., Hong, J., Feng, W.: Transparent Accelerator Migration in a Virtualized GPU Environment. In: 12th IEEE/ACM Symposimu on Cluster, Cloud and Grid Computing (CCGrid), pp. 124–131. IEEE Press (2012)

[5] Alécio, P.D.B., Carlos, E.P., Arjan, K., Andre, S., Dieter, W.F.: An effective dynamic scheduling runtime and tuning system for heterogeneous multi and many-core desktop platforms. In: 13th IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 78–85. IEEE Press (2011)

[6] Alexander, H., Michael, K., Bungartz, H.: From GPGPU to Many-Core: Nvidia Fermi and Intel Many Integrated Core Architecture. IEEE Science & Engineering Magazine 14(2), 78–83 (2012)

[7] Top500 supercomputer sites, `http://www.top500.org/blog/lists/2013/11/press-release`

[8] Jeffrey, S.V., Richard, G., Jack, D., Karsten, S., Bruce, L., Stephen, M., Jeremy, M.: Keeneland: Bringing heterogeneous gpu computing to the computational science community. IEEE Science & Engineering Magazine 13(5), 90–95 (2011)

[9] Fan, K., Kudlur, M., Dasika, G., Mahlke, S.: Bridging the computation gap between programmable processors and hardwired accelerators. In: 15th IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 313–322. IEEE Press (2009)

[10] Givargis, T., Vahid, F.: Platune: A tuning framework for system-on-a-chip platforms. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (CADICS) 21(11), 1317–1327 (2002)

[11] Tan, Z., Waterman, A., Avizienis, R., Lee, Y., Cook, H., Patterson, D., Asanović, K.: RAMP gold: An FPGA-based architecture simulator for multiprocessors. In: 47th ACM Design Automation Conference, pp. 463–468. ACM Press (2010)

[12] Intel developers guide, `http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-coprocessor-system-software-developers-guide.html`

[13] Diaz, J., Camelia, M., Alfonso, N.: A survey of parallel programming models and tools in the multi and many-core era. IEEE Transactions on Parallel and Distributed Systems (TPDS) 23(8), 1369–1386 (2012)

[14] Saule, E., Umit, V.C.: An early evaluation of the scalability of graph algorithms on the Intel MIC architecture. In: 26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), pp. 1629–1639. IEEE Press (2012)

[15] Marjan, M., Jan, H., Anthony, M.S.: When and how to develop domain-specific languages. ACM Transactions on Computing Surveys (CSUR) 37(4), 316–344 (2005)

[16] Michael, D.L., Jamison, D.C., Wang, H., Meng, T.H.: Merge: A programming model for heterogeneous multi-core systems. ACM Transactions on SIGOPS Operating Systems Review 42(2), 287–296 (2008)

[17] Naila, F., Andrew, K., Gregory, D., Sudhakar, Y., Karsten, S.: A framework for dynamically instrumenting gpu compute applications within gpu ocelot. In: 4th ACM Workshop on General Purpose Processing on Graphics Processing Units, pp. 9–17. ACM Press (2011)

[18] Arvind, S., Lee, H., Brown, K., Rompf, T., Chafi, H., Wu, M., Atreya, A., Odersky, M., Olukotun, K.: OptiML: An implicitly parallel domain-specific language for machine learning. In: 28th IMLS International Conference on Machine Learning (ICML), pp. 609–616. IEEE Press (2011)

[19] Gelado, I., Stone, J.E., Cabezas, J., Patel, S., Navarro, N., Hwu, W.: An asymmetric distributed shared memory model for heterogeneous parallel systems. ACM Transactions on SIGARCH Computer Architecture News 38(1), 347–358 (2010)

[20] Yang, Y., Xiang, P., Kong, J., Zhou, H.: A GPGPU compiler for memory optimization and parallelism management. ACM Sigplan Notices 45(6), 86–97 (2010)

[21] Hameed, R., Qadeer, W., Wachs, M., Azizi, O., Solomatnikov, A., Lee, B.C., Richardson, S., Kozyrakis, C., Horowitz, M.: Understanding sources of inefficiency in general-purpose chips. In: 37th IEEE/ACM International Symposium on Computer Architecture (ISCA), pp. 37–47. IEEE Press (2010)

[22] Qin, S., Geng, X., Jiang, Y.: Automatic Dynamic Task Distribution between CPU and GPU for VR Systems. Applied Mechanics and Materials 157, 1324–1330 (2012)

[23] Augonnet, C., Thibault, S., Namyst, R., Wacrenier, P.A.: StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. Concurrency and Computation: Practice and Experience (CCPE) 23(2), 187–198 (2011)

[24] Winter, J.A., Albonesi, D.H., Shoemaker, C.A.: Scalable thread scheduling and global power management for heterogeneous many-core architectures. In: 19th ACM International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 29–40. ACM Press (2010)

[25] Song, H., Choi, K.: Autonomic Diffusive Load Balancing on Many-core Architecture using Simulated Annealing. In: 9th International Conference on Autonomic and Autonomous Systems (ICAS), pp. 90–95. IEEE Press (2013)

[26] Bartzas, A., Bellasi, P., Anagnostopoulos, I., Silvano, C., Fornaciari, W., Soudris, D., Melpignano, D., Ykman-Couvreur, C.: Runtime Resource Management Techniques for Many-core Architectures: The 2PARMA Approach. In: The International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA), pp. 835–840. IEEE Press (2011)

Load Balancing in MapReduce Based on Data Locality

Yi Chen[1], Zhaobin Liu[1,⋆], Tingting Wang[1], and Lu Wang[2]

[1] School of Information Science and Technology,
Dalian Maritime University, Dalian, 116026, P.R. China
zhbliu@gmail.com
[2] China Academy of Civil Aviation Science and Technology,
Beijing, 100028, P.R.China

Abstract. With explosive growth in data size at era of information, MapReduce - a programing mode, which can process data in parallel, has been widely used. However, the original system gradually exposes some shortcomings. For example, handling skewed data can cause the imbalance of the system loads. After mapper processes data, the result will be sent to reducer by partition function. An inappropriate partition algorithm may result in poor network quality, the overloading of some reducers and the extension of the execution time of job. In summary, using an inappropriate algorithm to process skewed data will form a negative impact on the system performance. In order to solve load imbalance problem and improve performance of cluster, we plan to design an effective partition algorithm to guide the process of assigning data. Therefore, we develop an algorithm named CLP - Cluster Locality Partition, this algorithm consists of three parts: Preprocess part, Data-Cluster part and Locality-Partition part. The experimental results illustrate that the algorithm proposed in this paper is better than the default partition algorithm in the aspects of execution time and load balancing.

1 Introduction

With rapid growth of information and data, it becomes more and more important for companies and research institutions to possess a computing system that can analysis and process large-scale data quickly and efficiently. In this background, MapReduce[1] has been well known by more and more people. MapReduce, which presented by Google Company, is a popular programming model for parallel processing massive data set, and it has been used to multi-field and various purposes[2][3][4][5].

Hadoop is an excellent open-source implement platform about MapReduce, such as Facebook and The New York Times are using the platform. MapReduce consists of two stages - Map stage and Reduce stage, data can be processed on the two phases in parallel, so it can effectively reduce the runtime of processing large-scale amounts of data. However, MapReduce is not perfect.

⋆ Corresponding author.

This method may cause the imbalance of the system loads in some applications, especially when skewed data are processed. Skewed data appears not only on the map phase but also on the reduce phase, and can weaken the overall performance of the system. In this paper, we propose to use a partition algorithm to weaken the impact of skewed data on the performance of the system. The process of partition is that transfer the result from mapper to corresponding reducer. The common partition method is applying a hash function to each (key, value) pair and assigning a partition number to each pair. The default hash function in Hadoop is Hash(HashCode(intermediate key)mod numReducer). This hash function is efficient when keys equally appear and uniformly distribute. In this situation, reducers can receive almost the same number of key, valuepairs, and if reducers own same processing capacity, the runtime of each reducer will almost same. However, it is not common situation, while the data skew appears. Like PageRank and join operation, these applications often need to process skewed data. Therefore, the data cannot be evenly distributed to various nodes. Thus it is easy to have some negative impacts on the performance of clusters. For example, the amount and time of data transmission in the network and the reducers' loads are increased, and the execution time of job is extended. Therefore, it is necessary to use an effective and efficient partition algorithm to assign data. In this paper, firstly, we describe the reason why skewed data causes performance degradation of MapReduce and then, we present a partition algorithm named CLP. The CLP partition algorithm includes three main parts. The first part is preprocess part. In this part, we use random sampling to analyze input data and use an extra MapReduce job to gather the information we need. The second part is data-cluster part. This part is to form multiple data-clusters with all data. The data with the same key will be put into the same data-cluster. The size of these data-clusters is similar and their number is the same as that the reducers. The final part is locality-partition that assigns data-clusters to suitable process node according to data locality. We had tested the CLP partition algorithm, and it had been proved that the CLP could fix the issue that skewed data caused and improved performance of cluster very well.

The rest of this paper is organized as follows. We describe the problem which skewed data caused in detail and take an example in section II. SectionIII talks about the related work and the solutions about skewed data. In sectionIV, we present our CLP partition algorithm in detail. Then we use CLP in a real application and show the experiments result in sectionV. Finally, we give a conclusion in sectionVI.

2 Issue Description

2.1 MapReduce Model

There are two main stages in MapReduce - Map and Reduce. In the Map stage, data is translated into (key, value) format first. Then all data pairs (key, value) are processed in parallel by Map function and output intermediate values. Finally, all values associated with the same key will be sent to a same reducer by partition function, this process can be called shuffle. In the reduce stage, every reducer takes partition data as input, and performs the reduce function. Then reducers output the final results.

Map stage and Reduce stage are connected to the partition part, an important factor that affects the performance of system. The default partition algorithm can produce system load imbalance when encounters skewed data[6], because it only considers key, not their size. The default partition function can equally distribute keys, so every reducer can process the same number of keys. However, the input of each reducer is not only key, but a data pair like (key, value) and the number of values of each key is different, so some reducers may process more data though keys are equally assigned to reducer. This causes load imbalance of reducer. In addition, inappropriate partition increases data-transfer volume and extends transmission time. The output of mapper is sent to the corresponding reducer according to partition function. Three different locations of node afford the output of mapper. First, output can be processed at the node where mapper located. Second, the node that locates in same rack with mapper can be selected. Third, transmit to the node that locates in different rack with mapper. If the first condition cannot meet, there is no doubt that this increases the load of network and the time span of transmitting. Therefore, it is necessary to design a good partition algorithm.

2.2 Motivation Example

In Fig. 1, there are six keys with data are waiting for processing. The data size of each key is key_1: 6, key_2: 8, key_3: 10, key_4: 13, key_5: 20 and key_6: 30 respectively, so the sum of data size is 87. It is load balancing in map part, so each mapper processes 29 values. After default partition, keys can be evenly distributed to 3 nodes, node 1 processes data set of key_1 and key_4, node 2 processes data set of key_2 and key_5, and the rest data is processed by node 3. Obviously, the data size that in node 3 is bigger than other nodes. This reveals load imbalance among reducers and leads to node 3 to be a lagging reducer, so it prolongs the runtime of the whole job. We use function 4 to measure data locality, so the data locality about node 1 is 13.8 %, thus 86.2% data transmits from other node or even in different rack, and so it occupies a mess of network resource and extends transmission time.

3 Related Work

People have a high requirement against timeliness and resource utilization about Map-Reduce, so load imbalance ignored before has been paid more and more attention. Kwon et al. [7] made an intensive study of skew. They presented five types of skew, and proposed five practices to mitigate skew. In paper[8][9], they give solution to solve the data skew in join operation, but these methods cannot meet the demand in other applications. When using inappropriate partition algorithm, it can easily cause load unbalance problem among nodes. In the cluster,

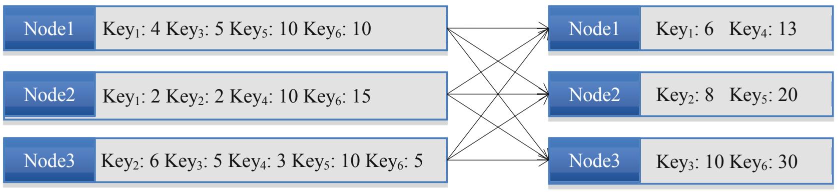

Node1
Key1: 4 Key3: 5 Key5: 10 Key6: 10
Node2
Key1: 2 Key2: 2 Key4: 10 Key6: 15
Node3
Key2: 6 Key3: 5 Key4: 3 Key5: 10 Key6: 5
Node1
Key1: 6 Key4: 13
Node2
Key2: 8 Key5: 20
Node3
Key3: 10 Key6: 30

Fig. 1. Issue Description. Key_x: y represent Key_x has y values.

some stragglers will appear. The problem had been described in[1], in this paper, Jeffrey Dean and Sanjay Ghemawat proposed the backup task that is used to alleviate the problem of stragglers. However, this method cannot perform well in heterogeneous environment. Therefore, Zaharia[10] proposed LATE, it defines fast/slow node and slow/fast task, and evaluates remain completion time of all task. The method is taking the task that has the longest remain time to fast node. All previous methods are passively to solve data skew after appearing load imbalance, which may result in a lot of issue. On one hand, finding a node to instead of straggler will lengthen the runtime. On the other hand, data is processed again when the task is assigned to a new node. It wastes previous computing resource. To solve this problem, SkewTune[11] was developed. It repartitions the unprocessed input data of a task with the greatest expected remaining processing time to an idle node and fully utilizes it. It can achieve good load balance, but dynamic load balancing method needs more system resource. In addition, network is transmission media from mapper to reducer, so the network load is also important essential should to be considering[12][13][14]. Faraz Ahmad proposed Tarazu[15] algorithm that focuses on network load and reducer load. There are three parts in Tarazu, 1. CALB is used to estimate network load. 2. CAS is used to avoid bursty network traffic. 3. PLB to distribute data to reduce node according to computing power of reduce node. The shortage about Tarazu is that making process of map discontinuously. It also wastes system resource.

4 The CLP Algorithm

4.1 Design Overview

The default partition algorithm is suitable for the condition that keys equally distribute and their data size is almost same. If the data set cannot meet this condition, it is easily to make network congestion and load imbalance. In order to fix these issues, we develop CLP partition algorithm. The CLP partition algorithm uses a three-parts partition algorithm to handle data skew. The three stages are preprocess, data-cluster and locality-partition, and it needs to run two MapReduce phases. The CLP partition algorithm can improve cluster performance in two aspects, one is reducing data volumes that transmit in network, and the other is balancing reducer load. We improve network condition based on data locality, it means that try best to let immediate data stay in original process node. It not only reduces the transmission data volume, but also shortens the transmission time. In order to balance system load, the CLP combines the data set of keys into a number of larger data-clusters, then sends these data-cluster to reducers. The data size of every data-cluster is nearly equal, so every reducer processes the nearly equally data volume. It can achieve our goal about load balance of each reducer.

4.2 CLP Implementation

Preprocess. This is an important part in CLP partition algorithm. In this part, we use random sampling to sample input data, and get the data distribution information by using extra MapReduce. The process is shown in Fig. 2. This part is used to help make partition.

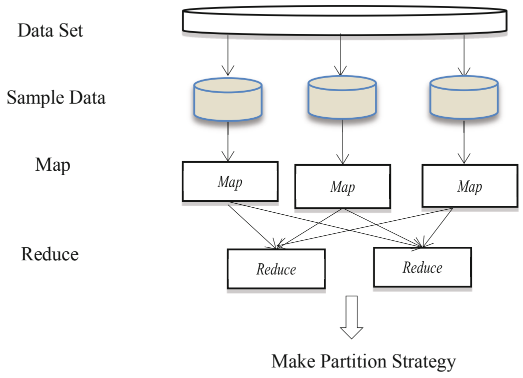


Fig. 2. Extra MapReduce about Sampling

Sampling is used to choose a representative sample from a target collecting data. It can help us understand the distribution of the overall data after the completion of analysis of only a fraction of the data, so it can reduce analysis time. At first, we get the sample by random sampling. Then we use an extra MapReduce job to analyze the sample data. Finally, we get the information about data distribution. The preprocess starts before practical MapReduce application. The CLP partition algorithm is proposed base on this information.

Using sampling instead of asynchronous to gather data distribution information has many advantages for Map and Reduce. In MapReduce operating mechanism, the execution of reduce phase begins after 5% of the map phase does (5 % is by default)[16]. Like[17], it used asynchronous Map and Reduce to get data distribution information. However, we choose using random sampling to analyze the data size. It doesn't break the MapReduce process, and doesn't need to wait for completing all Map tasks, because we use an extra MapReduce job to gather information before practical MapReduce application. In real MapReduce application, the intermediate (key, value) pairs can be immediately assigned to the corresponding reducer. Thus, it saves a lot of waiting time to reducer. We just need to assign a little resource for sampling MapRedcue job, which compare with the resource about original MapReduce system consumes when process skewed data is very little. Therefore, sampling can be used to help make a partition algorithm.

Although random sampling cannot give us very accurate data distribution information compare with real data, the sampling error is limited. We use the unbiased estimator to prove its reasonability, because the unbiased estimator is often used to evaluate sampling. We assume that Set A is the whole data, which is sampled by our extra MapReduce job. In this paper, the whole data is N keys. We do A(k) that presents Set A Bernoulli experiments. Set B is a sample which sampling from the whole data. Each key has the same probability (denoted by p) of being chosen in sample. Every key in Set A may be selected by sampling, it means every key will appear in Set B with probability p, so every key has two choices, key is selected or is not selected. This condition meets binomial distribution. We can use formula 1 to present, and use S(k) to present the process of sampling set B experiments.

$$S(k) \sim B(n, p) \tag{1}$$

If $\widehat{A}$(k) satisfies formula 2, it means that $\widehat{A}$(k) is unbiased estimator.

$$E(\widehat{\theta}) = \theta \tag{2}$$

The process of proof is as follow.
$\widehat{A}$(k) $=\frac{1}{p}S(k)$, so E($\widehat{A}$(k))=E($\frac{1}{p}$S(k))=$\frac{1}{p}$ E(S(k)) $=\frac{1}{p}$ A(k)p=A(k). Therefore, we can get a relatively satisfying result.

Data-Cluster. Based on the sampling result, we can gather the approximation data size (the number of values) of a key. Then we use this information to make partition algorithm. In this section, we assume that the performance of every node is same, and every node starts their task at the same time. The node which has largest data need to process decides the finish time. Uniform distribution data to each node will fasten the completion time. We use formula 3 (Impact Factor) to show how each key and their values impact nodes. The greater the SP, the greater impact on load, and the more processing time it takes the process node.

In MapReduce, all (key, value) pairs that associated with same key will be sent to one reducer by partition algorithm, and the number of (key, value) pair decides data size which need to process. The default partition algorithm is using a hash function, it just considers how to distribute equally keys while ignores the data size of each keys. So it can cause load imbalance among reducers. In order to balance the load of each reducer when process skewed data, we design the partition algorithm consider both keys and the number of values. We design several big data-clusters whose quantity is same with reducer. These data-clusters are formed by combining some data set of keys, and data size of each data-cluster is almost same. Then one data-cluster is sent to one reducer. Therefore, it can balance the load of reducer, because the size of data-cluster is nearly same. However, it is NP-Hard problem to combine data into data-clusters and make every data-cluster have similar data size. It is expensive to obtain the optimal assignment. Therefore, we provide a heuristic partition method to form several data-clusters; it can achieve sub-optimal load balancing. There are three steps in data-cluster part. At first, sort keys in descending order base on SP_i, we use SP_i to represent the impact to load imbalance. The key with the biggest SP_i is selected at first, and it is send to the smallest data-cluster. The smallest data-cluster means that the sum of SP_i that has been put in is smallest. The data-cluster part algorithm is as shown in Alg. 1.

$$ImpactFactor : SP_i = \frac{key_iCount}{T/RCount} \tag{3}$$

RCount is the number of reducer. T indicates total number of data:(0<i<the number of Key). The range of SP is SP≥0. As SP_i with small value has a little impact on processing time, so we sort the keys in descending order according to their impact factor.

Algorithm 1. CLP Algorithm: Data-Cluster Part

```
Input:
    KSP={$(Key_1, SP_1), (Key_2, SP_2) \ldots (Key_x, SP_x)$};
    RP={$RP_1, RP_2, , RP_y$};
    $y$=the number of reduce node
    $RP$ indicates data-cluster
    Initial value=0
    One data-cluster send to one reduce node
Output:
    $RP_{new}$={$RP_1, RP_2, \cdots, RP_y$}
1. Sort $SP_x$            //Sort SP in descending order
2. $i = 1$
3. while $i \le x$ do
4.    Sort($RP_y$)            //Sort $RP_j$ in ascending order
5.    Select$(Key_i, SP_i)max \to (RP_y)min$
6.    Calculate($RP_y$)          //$RP_y = \sum SP_Z$
7.    $i \leftarrow i + 1$
8. end while
```

Locality-Partition. The data locality is determined by the storage locations of task and the data. There are three levels about how good the locality is. The locality is best when the locations of the task and the data are on the same node. The locality is average when their locations are on the same rack, not on the same node. The locality is worst when their locations are not on the same rack. The common rule is that transmit task to the node prefer to transmit data. The reason is task is smaller than data, so transmit task will reduce wasting network resource. In Map stage, the computing system uses the method. However, in reduce part, the system has to transmit data to reduce node, so cannot promise data locality. Therefore, we hope use the character of data locality to make most data leave in the original process node, so it just needs to transmit a small part of data. So, it cannot only shorten network delay, but also reduce occupying network resource.

After data-cluster part, the step is that assign data-cluster to appropriate reduce node. Chose an appropriate node can reduce communication traffic and reduce network transmission time. In our partition algorithm, partitioning data-cluster to reducer is based on data locality, and we use formula 4 to measure the data locality.

$$KL_i^j = \frac{Key_i^j Count}{\sum_{i=1}^{j} Key_i^j Count} \tag{4}$$

$Key_i^j Count$ indicates the number of values in Map node j. indicates total number of data in node j. The bigger value of means better data locality. Then select the best node according to formula 5, the data-cluster will be sent to the reducer with the biggest KR values.

$$KR_j = \sum (SP_i * KL_i^j) \tag{5}$$

The locality-partition part is presented in Alg. 2.

CLP Example. In Fig. 3, it shows that the improvement and result when using CLP partition algorithm to process skewed data. It uses same data with issue example. At first, using random sampling and an extra MapReduce job analyze data distribution information. If we ignore sampling error, we can get accurate information about keys and their values size. Then it will be formed some data-clusters according to the data-cluster part of CLP. In our example, it forms three data-clusters, we use c_1, c_2 and c_3 to represent these data-cluster. One data-cluster sends to one reducer. It is obviously shown that key_1, key_3 and key_4 make up c_1, key_2 and key_5 form another data-cluster c_2 and key_6 will in data-cluster c_3. Therefore, each data size of data-cluster is 29, 28 and 30. Although the keys cannot equally appear among reducer, the values that each reducer needs to process is very closely. It is more fairness to distribute data to reducer than before, so data-cluster can balance load of each reducer. Finally, these data-clusters will be partitioned to suit reducer according to KR_j. We partition data-cluster that has max total value of KR_j. It is easily find that the max value is 50% of c_3, so c_3 is sent to reducer2 at first. In same procession,

Algorithm 2. CLP Algorithm: Locality-Partition Part

```
Input:
    RP={$RP_1, RP_2, , RP_y$};
    $R_y$: Reduce node set
Output:
    RP={$RP_y, R_y$}
1.  Calculate $KL_i^j$
2.  Sort $RP_y$            //Sort $RP_y$ in descending order
3.  $i = 1$
4.  while $i \leq y$ do
5.      Calculate $KR_j$          //SP from data-cluster RP
6.      Sort($KR_j$)          //Sort $KR_j$ in descending order
7.      $RP_y \rightarrow R_y(max\ KR)$          //PR
8.      Remove node $R_y$
9.      $i \leftarrow i + 1$
10. end while
```

c_1 is assigned to reducer1 and c_2 is partition to reducer3. The worst value of data locality is 31% and this value is better than the best data locality in issue example, so it is proved that using CLP reduces the volume of transmission data. In summary, the CLP partition algorithm meets our requirement, and it can fix the issue which data skew caused.

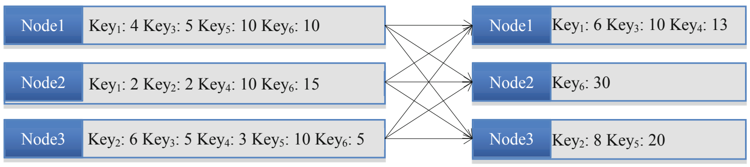

Node1
Key1: 4 Key3: 5 Key5: 10 Key6: 10
Node2
Key1: 2 Key2: 2 Key4: 10 Key6: 15
Node3
Key2: 6 Key3: 5 Key4: 3 Key5: 10 Key6: 5
Node1
Key1: 6 Key3: 10 Key4: 13
Node2
Key6: 30
Node3
Key2: 8 Key5: 20

Fig. 3. CLP Algorithm Example

5 Performance Evaluation

Our test bed runs on Ubuntu 10.10 Server and Cloudera Hadoop 0.20.2. The test bed contains 12 slave nodes and 1 master nodeand each node has two AMD Opteron 2212 2.00GHz CPUs, 8GB RAM, and 80GB HDD.

To evaluate CLP against default Hadoop partition algorithm (DP), we use the WordCount benchmarks without combiner to process real files. The processed data size of files varied from 1GB to 8GB. We evaluate the CLP partition algorithm from three aspects: 1. Execution time. It is the most basic measurable standard for a good algorithm, because the purpose of developing algorithm is reducing processing time. 2. Data locality. We want to reduce the transmit data and shorten transmission time by using CLP. Better data locality means less data will be transmitted in network, so we use data locality to judge the improvement of network condition. 3. Skew degree. It is used to judge the fairness of the distributing data. We use the standard deviation to measure data skew. The less value of standard deviation means the better fairness.

5.1 Execution Time

As shown in Fig. 4, the execution time of two algorithms increases with the growth of the data size. The reason is the larger the amount of data, the longer the execution time of task, it doesn't relate to algorithm. Compare DP and CLP, when data size is 1GB, CLP spends more time than DP. The reason is that the data skew is not obvious (data size of each key is closed) when data size is 1GB. So the default partition function can nearly equal distribute data to reducer like using CLP algorithm. Besides, the CLP need preprocess to analyze data, so it increases the process time. Therefore, CLP spends more time. However, CLP can decrease the process time when data size from 2GB to 8GB against DP algorithm. The phenomenon of data skew is more and more obvious, the shortages of default partition algorithm gradually exposed. Default partition algorithm cannot make sure that each reducer process equal data size of data, so some reducers process bigger data size than others. It makes load imbalance among reducers and produces some lagging reducers, and these lagging reducers spend more time finishing task. Therefore, the completion time of a whole job will be prolonged. While the CLP can fix issue that skewed data caused. It can analyze the data size of each key, and balance load of reducers. So it can efficiently reduce the process time of whole job.

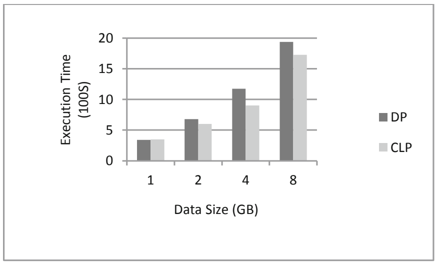


Fig. 4. Execution Time

5.2 Data Locality

Fig. 5 and Fig. 6 show the data locality about using different partition algorithms. We regard the standard deviation as standard of judgment. As is shown in Fig. 5, the average value of data locality that comes from 12 reducers. When data size is 1GB, we can find that DP can achieve good data locality. The reason is that data size of each key is small and closed, so each reducer can be assigned nearly equal data size. It takes a little influence on partition result. Therefore, it is closed to data locality of using CLP. But data locality of CLP is still better. When data size reach 2GB, the data locality of DP is very bad. It exposes the shortage when using default partition algorithm to process skewed data. While the CLP partition algorithm shows its outstanding performance. Comparing the data locality between DP and CLP, we can find that DP improves the data locality a lot. We can get same result at 4GB and 8GB data size, and the best data locality appears in 4GB. Better locality data signifies more data stays in original process node and is processed. It proves that data-clusters are assigned to right reducer, so more data stays in original node and data size that transmitted in network is reduced, thus transmission time is shortened. In addition, we make a presentation about data locality of 12 process node between DP and CLP when data is 4GB. As shown in Fig. 6, it is easily to find that the highest value of data locality is beyond 60% by using CLP partition algorithm, but the highest value of data locality in DP partition algorithm is only beyond 25%. Besides, the lowest value of data locality in CLP is also more than the highest value of DP, it is very clear that CLP partition algorithm can make more data stay in original process node to be processed, so it can decrease the transmission data among nods during shuffle phase. In a summary, the CLP partition algorithm can help improve the network condition of whole cluster.

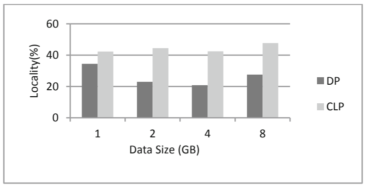


Fig. 5. Data Locality

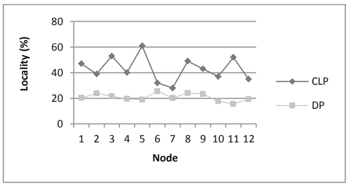


Fig. 6. Locality of each Node

5.3 Skew Degree

As is shown in Fig. 7, in the smallest data set, the data skew degree between DP and CLP are closed, because the number of values of each key is much closed, keys can nearly equally be distributed to reducer by DP, so reducers process

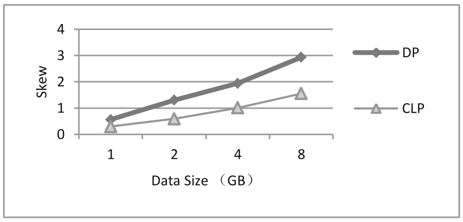


Fig. 7. Skew Degree

nearly equally data size of input. However, with the growth of data skew, the CLP can improve the data skew degree more and more efficient. While it also shows a trend of weakening the improvement, because some keys with very big data affect the performance of CLP.

6 Conclusions

In MapReduce, network condition and system load are very important factors affecting the performance of cluster. In this paper, we show the issue that skewed data caused, and present a partition algorithm that based on data locality. The CLP partition algorithm consists of three parts. In preprocess, we use random sampling to gather useful data information. Data-Cluster part is in order to make sure balancing load of each reducer. Locality-Partition part is used to partition data to right reducer. Three parts work together can solve the problem that skewed data caused, so it can improve the cluster performance well. In the future work, we intend to conduct more experiments to evaluate CLP with different applications and take a comprehensive study on how to perfect our partition algorithm.

Acknowledgements. This work is supported by the National Science Foundation for Distinguished Young Scholars of China under grant no. 61225010, NSFC under grant nos. of 61370198, 61370199 and 61300187, Program for New Century Excellent Talents in University (NCET-10-0095) of Ministry of Education of China, The Fundamental Research Funds for the Central Universities under grant no. of 3132014215, and China Scholarship Council Program.

References

1. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51, 107–113 (2008)
2. Morton, K., Balazinska, M., Grossman, D.: Paratimer: A progress indicator for mapreduce dags. In: Proceedings of the, ACM SIGMOD International Conference on Management of Data, pp. 507–518. ACM (2010)
3. Ferreira Cordeiro, R.L., Traina Junior, C., Machado Traina, A.J., López, J., Kang, U., Faloutsos, C.: Clustering very large multi-dimensional datasets with mapreduce. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 690–698. ACM (2011)
4. Li, B., Mazur, E., Diao, Y., McGregor, A., Shenoy, P.: A platform for scalable one-pass analytics using mapreduce. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 985–996. ACM (2011)
5. He, B., Fang, W., Luo, Q., Govindaraju, N.K., Wang, T.: Mars: A mapreduce framework on graphics processors. In: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, pp. 260–269. ACM (2008)
6. Gufler, B., Augsten, N., Reiser, A., Kemper, A.: Load balancing in mapreduce based on scalable cardinality estimates. In: 2012 IEEE 28th International Conference on Data Engineering (ICDE), pp. 522–533. IEEE (2012)

7. Kwon, Y., Balazinska, M., Howe, B., Rolia, J.: A study of skew in mapreduce applications. Open Cirrus Summit (2011)
8. Xu, Y., Kostamaa, P., Zhou, X., Chen, L.: Handling data skew in parallel joins in shared-nothing systems. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1043–1052. ACM (2008)
9. Xu, Y., Kostamaa, P.: Efficient outer join data skew handling in parallel dbms. Proceedings of the VLDB Endowment 2, 1390–1396 (2009)
10. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R.H., Stoica, I.: Improving mapreduce performance in heterogeneous environments. In: OSDI, vol. 8, p. 7 (2008)
11. Kwon, Y., Balazinska, M., Howe, B., Rolia, J.: Skewtune: mitigating skew in mapreduce applications. In: Proceedings of the, ACM SIGMOD International Conference on Management of Data, pp. 25–36. ACM (2012)
12. Vahdat, A., Al-Fares, M., Farrington, N., Mysore, R.N., Porter, G., Radhakrishnan, S.: Scale-out networking in the data center. IEEE Micro 30, 29–41 (2010)
13. Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., Stoica, I.: Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European Conference on Computer Systems, pp. 265–278. ACM (2010)
14. Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: Portland: A scalable fault-tolerant layer 2 data center network fabric. ACM SIGCOMM Computer Communication Review 39, 39–50 (2009)
15. Ahmad, F., Chakradhar, S.T., Raghunathan, A., Vijaykumar, T.: Tarazu: Optimizing mapreduce on heterogeneous clusters. ACM SIGARCH Computer Architecture News 40, 61–74 (2012)
16. Hammoud, M., Sakr, M.F.: Locality-aware reduce task scheduling for mapreduce. In: 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), pp. 570–576. IEEE (2011)
17. Ibrahim, S., Jin, H., Lu, L., Wu, S., He, B., Qi, L.: Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 17–24. IEEE (2010)

RD-PCA: A Traffic Condition Data Imputation Method Based on Robust Distance

XueJin Wan, Yong Du, and Jiong Wang

Beijing Transportation Information Center, Beijing, China
{wanxuejin,duyong,wangjiong}@bjjtw.gov.cn

Abstract. Dynamic Transportation Information Service has penetrated into residents' travels. The current problems that transportation information services face are variable such as real-time traffic forecasting, traffic managing and traffic induction. The above problems are related to the quality of historical traffic condition data. Due to a limited of GPS data collecting, the collected GPS data which scarcely covers the whole road network leads to incomplete and error traffic condition data. In consequence, two serious problems of traffic condition data quality manifest in incompleteness and low accuracy. This paper extends RD-PCA method which preliminarily focuses on the accuracy of imputing to prevent the estimating results from being impacted by outliers and aims at guaranteeing the completeness of imputing. The method excludes error data taking data quality measurement criterions. By adopting a measure factor, this method detects outliers and standardizes them, then constructs a robust feature space and imputes the missing data. The experimental results show that the proposed method can guarantee a high completeness and high accuracy under the condition of different missing rates.

Keywords: Imputing missing data, robust feature space, outliers detecting, accuracy.

1 Introduction

Dynamic transportation information service (TIS) has penetrated into many aspects of citizens' travel. The more travel demands arise, the more demands for good transportation information services will be produced. Current problems that transportation information services face are variable such as real-time traffic forecasting, traffic managing and traffic induction[1]. The above problems are related to historical traffic condition data quality. High-quality traffic condition data can outstandingly supplement transportation information services. However, due to a limited number of GPS data collecting devices, as well as accidental data transmission failure and imperfect data processing systems, collected GPS data which scarcely cover the whole road network leads to incomplete and error traffic condition data. These data are output by traffic condition data computing system (TCDCS) [3]. In consequence, two serious traffic condition data quality problems are represented as incompleteness and low accuracy. The problems are urgent to be resolved to support superb TIS.

Some notable imputation methods include historical mean method(Li L et al.2013), Kalman filter method(Guo J et al. 2014), time series modeling(Tan H et al.2013) , non-parametric regression method (Chen C. et al 2003), classical PCA method(Qu L. et al. 2009), and PPCA method(Qu L. et al. 2009). These have been shown to perform well in their respective context, it turns out that their fatal defects in detecting and handling outliers will affect the accuracy of imputing and lead to another problem of data quality. In addition, they are not suited to impute missing values when the missing rate is high.

This paper extends a new effective method called Principal Component Analysis based on Robust Distance (RD-PCA) to detect outliers and impute missing values. RD-PCA method excludes error data by taking data quality measurement criterions and detects outliers by adopting a measure factor called robust distance. Missing values are finally imputed by constructing a robust feature space. Three experiments were conducted to analyze and discuss the accuracy and completeness of imputing. The experimental results show that RD-PCA is superior on accuracy compared with existing methods and can guarantee a low missing rate after imputing. Three main contributions can be made as below:

- **Continuous and intermittent missing values partition**. This method adopts three measurement criterions to exclude error data. To impute missing data, we partition missing values into continuous and intermittent missing values, and adopt different algorithms to impute them relying on their respective missing data features.
- **An algorithm to detect and standardize outliers**. The algorithm preliminarily adopts a measure factor to detect outliers and constructs a robust feature space to minimize the impacts of outliers. The method guarantees a high accuracy of imputing and completeness of imputing when the missing rate is not quite high.
- **A supplementary of high accurate and complete traffic condition data**. RD-PCA method has been used to impute missing data in application domain. It is proved to have the ability to support TIS by a supplementary of high accurate and complete traffic condition data.

The rest of this paper is organized as follows: In section 2, we summarize some previous works. In Section 3, we present the RD-PCA method. Section 4 reports experimental results and related discussions, and finally, we conclude in Section 5.

2 Related Work

There are plenty of imputing techniques proposed in literature (King et al.2001, Graham et al. 2003, Schafer and Kam 2003). Amount of missing data imputation methods have been proposed specifically for traffic missing data (Schafer J L, Graham 2002, Smith and Conklin 2002, Fabritiis and Ragona et al. 2008). Some methods specifically discussed for traffic condition data have attracted great attentions. These include historical mean method(Li L et al.2013), Kalman filter method(Guo J et al. 2014), time series modeling(Ye Q, et al.2012), non-parametric regression method (Chen C. et al 2003), classical PCA method(Qu L. et al. 2009), and PPCA method(Qu L. et al. 2009). Here we will only review some notable work due to space limitation.

Historical mean method (Du B et al. 2008) advocated for the first time is popular in that it imputes missing traffic condition data of one road-link with means of all known historical depending on traffic periodicity. This method can guarantee the completeness of imputing because outliers will be balanced as long as the amount of historical data is large enough. However, imputing values may deviate from ground truth values in consideration of that the contemporary traffic conditions are not alike but similar. Chen C. et al. (2008) put forward non-parametric regression-based method (NPR) which further imputes missing data by finding a data sequence from historical data sequences that proximately partly matches the current missing data sequence. Hence this method prefers to impute missing data directly from historical data rather than estimate an approximate value. However, the imputation method is terrible under normal traffic conditions. What's more, accumulating errors will exist when performing and the accuracy of imputing will be affected.

Although historical mean imputation method and non-parametric regression imputation method are frequently taken for missing value imputation, they both suffer from some defects. They neglect the fact that traffic flows may fluctuate significantly from day to day and contain stochastic variation within the same day. While the classical PCA-based method (Howard W. J. 2012) and PPCA-based method (Qu L. et al. 2009) consider an adaptive fusion of historical and in-a-day information, they outperform the historical imputation methods. CPCA method tries to explain the covariance structure of data by means of a small number of components. These components are linear combinations of the original variables, and often allow for interpretation and better understanding of different sources of variation. Unfortunately, both the classical variance and the classical covariance matrix are very sensitive to outliers. So the disadvantage of this method is that the first components may not capture the variation of the regular observations once outliers are presented.

Therefore, the above imputation methods above relying on historical traffic condition data have their fatal defects respectively. Estimate results of HM, CPCA and PPCA methods will be impacted severely by those outliers that cannot be neglected when adopting historical data. NPR method effectively adopts historical data in abnormal traffic conditions to minimize the gap between estimate values and ground truth values prominently. However, it performs terribly in imputing missing data under normal traffic conditions.

Hence, we are motivated to propose a new method to detect outliers effectively and improve accuracy and completeness of imputing. Afterwards, we conduct experiments to compare our proposed method against them.

3 RD-PCA Method

In this section, we firstly summarize the quality of original traffic condition data and classify abnormal data into error data, outliers and missing data. Then, three parts of this method are described which are extracting traffic condition data feature, excluding error data and imputing missing data in detail.

3.1 Traffic Condition Data Quality

Definition 1. Missing Rate ε [8] ε is defined as the degree to which missing values are presented in the attributes as required. Here we parametrically denote $\varepsilon = \frac{n_{imp}}{n_{total}} \times 100\%$, where n_{imp} represents number of missing values, and n_{total} represents number of measured values.

Traffic condition data quality can be affected by many uncertain factors such as broken GPS devices, accidental data transmission failures and abnormal data. In this paper, we classify abnormal data into error data, outliers and missing data.

Error Data. Suppose that GPS data collected by GPS devices can accurately reflect the traffic condition of one road. Due to defects of TCDCS, those outputting data which cannot reflect the traffic condition at that time are called error data. To prepare eligible data for imputing, error data have to be excluded from historical data with a measure criterion which is validity σ [8].

Outliers. Abnormal traffic conditions like traffic accidents, traffic congestions and traffic controls lead to data's deviation from those data in normal. These data are called outliers, which are output by TCDCS based on collected GPS data and accurately reflect traffic condition at that time.

Definition 2. Robust Distance[9]

Suppose that the original data matrix $X_0 = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ip} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} x_1^t \\ \vdots \\ x_i^t \\ \vdots \\ x_n^t \end{pmatrix}$, x_i^t, n and p represents respectively the ith vector, number of vectors and number of variables.

Then robust distance $d_i(x) = \sqrt{(x_i - med(x))^T (\mathrm{cov}(x))^+ (x_i - med(x))}, i = 1, 2, \cdots n$ [10], where $med(x)$ is the classical mean vector of x and $med(x) = (med(x_{j1}), med(x_{j2}), \cdots, med(x_{jp}))^T$, $j = 1, 2, \cdots n$.

If $d_i(x) \geq \chi^2_{0.025}(p)$ [10], x_i is an outlier, otherwise it is a normal one.

Missing Data. Due to a limited number of taxis and uneven distribution of taxis, not all the GPS data can be collected and traffic condition data cannot be obtained entirely through TCDCS based on the incomplete GPS data. These existing data which are not output by TCDCS are called missing data. In order to impute the missing part of data, here we divide missing data into two types: continuous missing values and intermittent missing values.

3.2 Extracting Traffic Condition Data Feature

Definition 3: **Road-link Missing Data Feature Area** of $link_0$

Suppose that $link_0$ is a link of road-network R_n, the 2nd adjacent road-link set[4] of $link_0$ is S^{l2}_{adj} .As shown in **Fig. 1**, assume that data of $link_0$ in m days are stored in D^T . The data of n timestamps in the i^{th} day are denoted as $(g_i)^T = (\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ij}, \cdots, \alpha_{in})^T, 1 \leq i \leq m, 1 \leq j \leq n$. $(g'_{k,i})^T = (\alpha_{k,i1}, \alpha_{k,i2}, \cdots, \alpha_{k,ij}, \cdots, \alpha_{k,in})^T$, $1 \leq k \leq m$, $1 \leq i \leq l$, $1 \leq j \leq n$ where $(g'_{k,i})^T$ represents data of n timestamps in the i^{th} day ,and l represents $Card(LS^2_{link_0})$. The traffic condition feature data are denoted as $D^T = (G_1, G_2, \cdots, G_k, \cdots, G_m)^T, (1 \leq k \leq m)$.

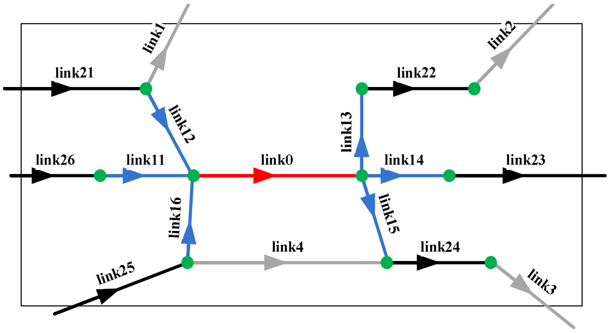


Fig. 1. Road-link Missing Data Feature Area of $link_0$

3.3 Excluding Error Data

Original traffic condition feature data matrix D^T consists of error data, outliers and missing data. To impute missing data, we have to exclude error data from the matrix by the following steps.

Step 1:Exclude error data by calculating stability μ [8] and validity σ [8] of $(g_i)^T$ and to construct D'^T .For each $\alpha_{ij} \in D'^T$ and $\beta_i \in D$, replace the inconformity values with 0 and reconstruct D'^T and D .

Step 2: Merge missing data segment described as $(g_0')^T = (\gamma_1, \gamma_2, \cdots; \gamma_q, \gamma_{q+1}, \cdots; \gamma_n)^T, (1<q<n)$ .Let $A = (\gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_q)^T \quad B = (\gamma_{q+1} \quad \gamma_{q+2} \quad \cdots \quad \gamma_n)^T, 1<q<n$, $G^T = (A^T \quad B^T)$. A and B are missing data segment and complete data segment. G represents the matrix of traffic condition data of $link_0$.

Step 3: Assume that X' is a data matrix of historical traffic condition data made up of historical traffic condition data of $link_0$ and its adjacent road-links. $X' = (g_1, g_2, \cdots, g_d, g_{1,1}', g_{1,2}', \cdots, g_{1,l}', \cdots, g_{m,1}', g_{m,2}', \cdots, g_{m,l}')^T$ .It is formalized as $X' = \left(X_A' \quad X_B'\right)$, where $X_B' = \left(f_1 \quad f_2 \quad \cdots \quad f_d \quad f_{l+1} \quad f_{l+2} \quad \cdots \quad f_{2l+1} \quad \cdots \quad f_{ml+1} \quad f_{ml+2} \quad \cdots \quad f_{(l+1)m}\right)$. $(X_A')^T$ consists of the first q vectors of $(g_{k,i}')^T$. $(X_B')^T$ consists of the last $(n-q)$ vectors of $(g_{k,i}')^T$.

After the above steps, data matrix X' is constructed by vectors of missing data, vectors of complete data in feature days and vectors of complete feature data of adjacent links.

3.4 Imputing Missing Data

After excluding error data, the process of imputing missing data consists of four parts: imputing intermittent missing data, detecting outliers, constructing robust feature space and imputing continuous data. The following section describes those methods in detail.

- **Imputing Intermittent Missing Data**

For intermittent missing data which can be described as $(g_0)^T = (\beta_1, \beta_2, \cdots; \beta_j, \cdots; \beta_n)^T, (1 \le j \le n)$,

we suppose that the value of β_j is missing, then $\beta_j = \dfrac{\beta_{j+1} + \beta_{j-1}}{2}$.

- **Detecting Outliers**

To detect outliers effectively, we regard robust distance $d_i(x)$ (mentioned in **Definition 2**) as a measure factor. Outliers are detected by gradually increasing calculation data points into data point set and extracting normal and outliers from the whole data set. The algorithm detects and standardizes outliers as below:

Step 1: Create an original data point set M_B of B (mentioned in 3.3), which contains m samples taking from X_B' , where $M_B = \{f_1, f_2, \cdots, f_s, \cdots, f_m\}, s = 1, 2, \cdots, m$. n and p represent row and column of X_B' .When $n > p$, $m = p+1$, otherwise $3 \le m \le 5$. Calculate $d_s(M_B, B), (s = 1, 2, \cdots, m)$ which is the distance between f_s and B ;

Step 2: Sort $d_s(M_B, B), (s = 1, 2, \cdots, m)$ in ascending order;
Step 3: Let $m = m+1$, if $m = n$, go to **Step 7**; otherwise go to **Step 4**;
Step 4: Sort f_s on basis of $d_s(M_B, B)$ in ascending order and create $M_B^{'}$, which is

$$M_B^{'} = \{f_1^{'}, f_2^{'}, \cdots, f_s^{'}, \cdots, f_m^{'}\}, s = 1, 2, \cdots, m;$$

Step 5: Create Q_B which contains the first k vectors of $M_B^{'}$, $Q_B = \{f_1^{'}, f_2^{'}, \cdots, f_i^{'}, \cdots, f_k^{'}\}, 1 \le i \le k$,where $k = \lceil \alpha \times m \rceil$. Calculate $d_s(M_B, B)$ again between f_s and B . $d_s(M_B, B) = \sqrt{(f_s - med(M_B))^T (\mathrm{cov}(Q_B))^+ (f_s - med(M_B))}$, where $med(M_B)$ is the mean of M_B and $\mathrm{cov}(Q_B)$ is the covariance deviation of Q_B ;
Step 6: Go to **Step 2**;
Step 7: If $d_s(M_B, B) \ge \chi^2_{0.025}(m, p)$, f_s is the outlier and add f_s to outlier set XN , otherwise add it to normal set XA .

Finally, $XN_{n_1 \times p}$ and $XA_{n_2 \times p}$ are constituted where $n_1 + n_2 = n$.

- **Constructing Robust Feature Space**[11]

To constitute robust feature space, the normal matrix $XA_{n_2 \times p}$ has to be standardized firstly. The operation is reducing the data space to the affine subspace. A convenient way to standardize it is to make a singular value decomposition of $XN_{n_1 \times p}$. The steps are described in detail as follows.

Step 1: We firstly get R descending eigenvalues after SVD, $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_i \ge \cdots \ge \lambda_p, 1 \le i \le rank(XN_{n_1 \times p})$, and calculate accumulating contribution rate[9] η of the first k values. k is meeting the demands only if

$$\eta = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{p} \lambda_j} \ge 90\%, 1 \le k \le rank(XN_{n_1 \times p}).$$

Step 2: Calculate the mean vector $(\mu_1, \mu_2, \cdots \mu_p)^T$ of $XN_{n_1 \times p}$ and standard deviation $(s_1, s_2, \cdots s_p)^T$ by $\mu_j = \sum_{i=1}^{n_1} XN_{ij}, s_j = \sqrt{\frac{\sum_{i=1}^{n_1} (XN_{ij} - \mu_j)^2}{n_1 - 1}}, 1 \le i \le n_1, 1 \le j \le p$

Step 3: Standardize $XA_{n_2 \times p}$ by $(\mu_1, \mu_2, \cdots \mu_p)^T$ and $(s_1, s_2, \cdots s_p)^T$. Standardized matrix can be represented as $Z_{n_2 \times p}$, so we can get $Z_{n_2,p}$ where

$$z_{ij} = \frac{XA_{ij} - \mu_j}{s_j}, 1 \le i \le n_2, 1 \le j \le p \,.$$

Z' is the first k (**Step 1**) vectors of S_{n_1,n_1} described as $Z'_{n_1 \times k}$. $Z_{n_2 \times k}$ consists of the first k vectors of $Z_{n_2 \times p}$. $ZA_{n \times k}$ can be described as $Z_{n_2 \times k}$ and $Z'_{n_1,k}$. So X_A' can also be standardized like this and complete data matrix $ZA_{n \times (q+k)}$. Finally, the robust data matrix is $Z = \begin{pmatrix} ZA_{n \times q} & ZA_{n \times k} \end{pmatrix}$.

- **Imputing Continuous Missing Data**

Suppose that the robust feature space matrix is Λ , then

$$\Lambda = \begin{pmatrix} G \\ Z \end{pmatrix} = \begin{pmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_q & \gamma_{q+1} & \gamma_{q+2} & \cdots & \gamma_n \\ Z_{1,1} & Z_{1,2} & \cdots & Z_{1,q} & Z_{1,q+1} & Z_{1,q+2} & \cdots & Z_{1,q+k} \\ Z_{2,1} & Z_{2,2} & \cdots & Z_{2,q} & Z_{2,q+1} & Z_{2,q+2} & \cdots & Z_{2,q+k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Z_{n,1} & Z_{n,2} & \cdots & Z_{n,q} & Z_{n,q+1} & Z_{n,q+2} & \cdots & Z_{n,q+k} \end{pmatrix} = \begin{pmatrix} A & B \\ Z_A & Z_B \end{pmatrix}$$

where Z_A consists of the first q column of Z and Z_B consists of the last k column of Z .Let $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_q)^T$, $B^T \approx \tilde{x}_1 \gamma_1 + \tilde{x}_2 \gamma_2 + \cdots + \tilde{x}_q \gamma_q$.

Then the problem of imputing the missing part transforms to that of resolving the best-fit curve through the least-square curve fitting algorithm [12].

Finally we can get A by $\tilde{x} = min_x \left\| Z_A x \text{-} Z_B{}^T \right\|_2$ and $A\tilde{x} = B$.

4 Experimental Results and Discussions

In this section, we propose a measurement criterion to verify the performance of our method. Afterwards, we conduct three experiments to verify the accuracy and completeness of RD-PCA method, and discuss the experimental results as well.

4.1 Performance Measurement Criterion

The accuracy of imputing is measured by Root Mean Square Error (RMSE)[8], which is a frequently adopted error measurement criterion reflecting an average performance of imputing. It is defined as $RMSE = \sqrt{\frac{\sum_{i=1}^{M}(t_{gro}^{i} - t_{est}^{i})^2}{M}}$, where M, t_{est}^{i} and t_{gro}^{i} are data quantities, estimated missing dataset and original dataset.

4.2 Comparison on Accuracy of Four Methods

Experiment 1 was done to make a comparison on accuracy of four imputing methods, i.e. HM, CPCA, NPR and RD-PCA. The experiment randomly selected a road link '59567200853' of Grade 2 located in 4th outer-ring. The missing rate of its traffic condition data was 1.77% on 3rd Dec, 2012. These data of 288 timestamps were evacuated in six kinds of missing rate, i.e. 10%, 20%, 30%, 40%, 50% and 60% respectively on basis of the changing trend of taxi quantities. **Fig. 2** manifests six results of evacuated data. In **Fig. 2**, each abscissa represents 288 timestamps and each ordinate represents speed of the link in that day. Red curves (points) stand for existing traffic condition values and blue curves (points) stand for the missing traffic condition values.

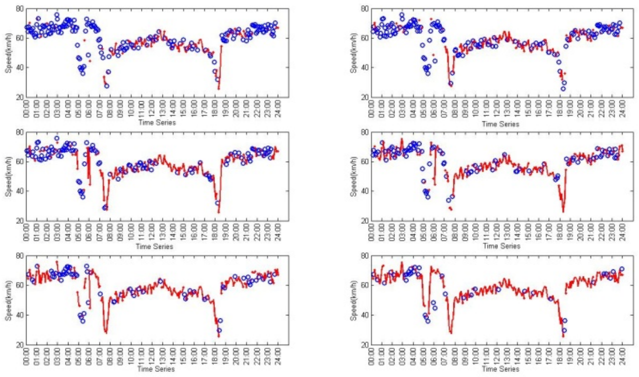


Fig. 2. Data distribution of different missing rate

We adopted the four methods above to impute six groups of missing data. The imputation results are shown in **Fig. 3**, which is made up of six blocks representing the estimating results of six missing rates respectively. The missing rates from 60% to 10% match six blocks respectively from the upper left one to the lower right one. In each block, the abscissa and ordinate represent 288 timestamps of the day and average speed of the link.

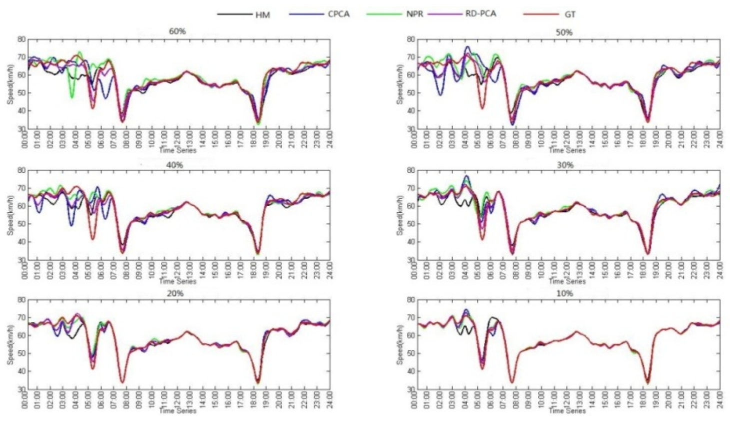


Fig. 3. Results of different missing rates(10%～60%)

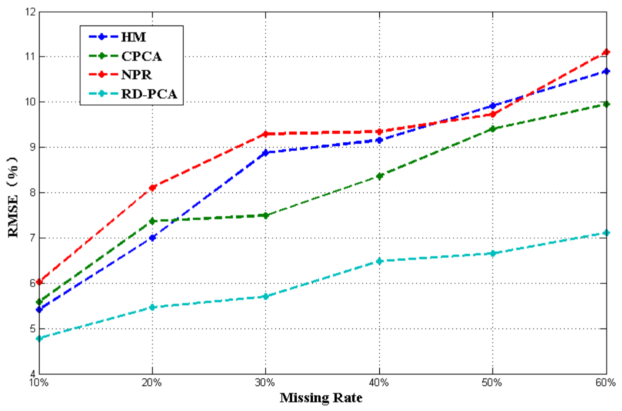


Fig. 4. RMSE of four methods

In accordance with the imputation result, RMSEs of imputation results were calculated and shown in **Fig. 4**. The results indicate that imputation accuracy of four methods varies with different missing rates. RD-PCA method is better than the other three methods whatever the missing rate is. In **Fig. 4**, the abscissa and ordinate stand for missing rate and RMSE. When missing rate is between 10% and 40%, NPR is the worst. The result also implies that when missing rate is higher than 50%, CPCA is better than HM and NPR. Besides, it is obvious that RMSE of our method varies at a certain curvature and it is still lower than 8% even if the missing rate is 60%.

Considering the experimental results of various missing rates, we confirm that RD-PCA method is superior to the other three. When the missing rate is low, four methods perform similarly. However, HM method performs terribly when the missing rate is high because it almost entirely neglects the specific features of traffic conditions. NPR method is impacted by accumulated error during the performance. We assume that the accumulated error amplifies along with increasing missing rate.

The result that RD-PCA method is better than CPCA method is attributed to that the feature space constructed by CPCA method will be impacted by outliers while RD-PCA method will not be.

4.3 Analysis on Imputation Completeness

In Experiment 2, 381 road-links of the 4th inner ring were selected to impute the missing data in the whole month of December, 2012. The experiment chose missing rates of 3rd Dec(Monday),4th Dec(Tuesday),7th Dec(Friday) and 8th Dec(Saturday) and then showed the distributions of missing rates before imputing in **Fig. 5**.

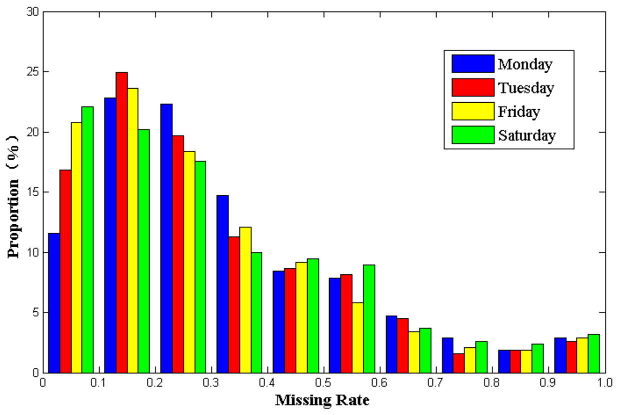


Fig. 5. Distribution of missing rates (before)

In this figure, the abscissa represents the missing rate which is divided into 10 sub-intervals defined as $RI_i = (0.1 \times i, 0.1 \times (i+1)], (i \in N, 0 \leq i < 10)$ .The ordinate represents proportion of number of links in each sub-interval.

The result implies that various missing rates affect completeness of imputing. **Fig. 5** shows that most missing rates of experimental links are between 10% and 40%. We adopted RD-PCA method to impute the missing data of all these road links. The missing rate distribution after imputing is shown in **Fig. 6**. The result indicates that the missing rate after imputing is lower than 5%.

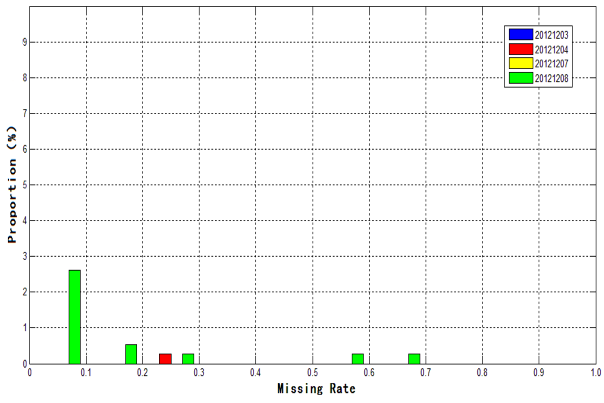


Fig. 6. Distribution of missing rates (after)

Despite that the completeness of imputing result is rather high, some data are still missing to some degree. Since missing rates of those data are approaching 100%, RD-PCA method fails to impute. The failure came out because that the historical data of those road links is too little to construct a robust feature space.

Furthermore, the result suggests that there is no direct relation between missing rates and feature days. **Fig. 5** indicates that in the same missing rate interval, quantities of links in different feature days are similar. The changing trend of each feature day is consistent with the missing rate.

5 Conclusions

In this paper, we have proposed a RD-PCA method to resolve the problems of traffic condition data quality of low accuracy and incompleteness. We firstly summarize the quality of missing traffic condition data, and classify abnormal data into error data, missing data and outliers. Secondly, we put forward a new method comprising extracting traffic condition data, excluding error data and imputing missing data. Afterwards, two experiments are conducted to analyze and to discuss the accuracy and completeness of imputing. The experimental results verify that RD-PCA method is robust and accurate to guarantee a low missing rate after imputing. However, the method does not perform well when the missing rate is approving 100%. Therefore, further studies have to focus on the completeness of imputing under the condition of high missing rate.

Acknowledgements. This work was supported by Beijing Science and Technology Development Funds (NO. D131100005113001).

References

1. Oki. What is VICS, http://www.oki.com/jp/SSC/ITS/eng/vics.html
2. Wen, Y.H., Lee, T.T., Cho, H.J.: Missing Data Treatment and Data Fusion Toward Travel Time Estimation For ATIS. Journal of the Eastern Asia Society for Transportation Studies 6, 2546–2560 (2005)
3. Graham, J.W., Cumsille, P.E., Elek-Fisk, E.: Methods for Handling Missing Data. Handbook of Psychology 2, 87–114 (2003)
4. Du, B., Xu, L., Ma, D., Lv, W.: Missing data compensation model in real-time traffic information service system. In: Fifth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2008, vol. 5, pp. 371–378. IEEE (2008)
5. Guo, J., Huang, W., Williams, B.M.: Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. Transportation Research Part C: Emerging Technologies 43(1), 50–64 (2014)
6. Li, L., Li, Y., Li, Z.: Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. Transportation Research Part C:Emerging Technologies 34, 108–120 (2013)
7. Qu, L., Li, L., Zhang, Y., et al.: PPCA-based missing data imputation for traffic flow volume: A systematical approach. IEEE Transactions on Intelligent Transportation Systems 10(3), 512–522 (2009)
8. Turner, S.: Dening and measuring traffic data quality: White paper on recommended approaches. Transportation Research Record: Journal of the Transportation Research Board 1870(1), 62–69 (2004)
9. Hubert, M., Rousseeuw, P.J., Branden, K.V.: ROBPCA: A new approach to robust principal component analysis. Technometrics 47 (2005)
10. Hubert, M., Van Driessen, K.: Fast and robust discriminant analysis. Computational Statistics & Data Analysis 45(2), 301–320 (2005)
11. Kumagai, M., Fushiki, T., Kimita, K., et al.: Spatial Interpolation of Real-time Floating Car Data Based on Multiple Link Correlation in feature space. In: Proc. of 13th World Congress of ITS, London, CDROM (2006)
12. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal 37(2), 233–243 (1991)
13. Wasito, I.: Least Squares Algorithms with Nearest Neighbor Techniques for Imputing Missing Data Values. Birkbeck College (2003)

Network-Aware Re-Scheduling: Towards Improving Network Performance of Virtual Machines in a Data Center

Gangyi Luo[1], Zhuzhong Qian[1], Mianxiong Dong[2], Kaoru Ota[3], and Sanglu Lu[1]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] National Institute of Information and Communications Technology, Japan
[3] Department of Information and Electronic Engineering, Muroran Insitute of Technology, Japan
`luogangyi@dislab.nju.edu.cn`, `{qzz,sanglu}@nju.edu.cn`, `mx.dong@nict.go.jp`, `ota@csse.muroran-it.ac.jp`

Abstract. An effective virtual machine allocation and scheduling algorithm can improve the utilization of physical servers, lower energy cost and improve the overall performance of datacenters. Current virtual machine scheduling algorithms mainly focus on the initial allocation of VMs based on the CPU, memory and network bandwidth requirements. However, Caused by finish of jobs or expiration of lease, related virtual machines would be shut down and leave the system which generate plenty of resource fragments. Such fragments lead to unbalanced resource utilization and the communication performance may decline significantly. This paper studied the network influence on some typical applications in datacenters and proposed a self-adaptive network-aware virtual machine re-scheduling algorithm to maintain an optimal system-wide status. Our algorithm had two stages. In the first stage, we checked whether re-scheduling was necessary and in the second stage perform a heuristic re-scheduling to lower communication cost among VMs. We use two benchmarks in a real environment to examine network influence on different tasks. To evaluate the advantages of the proposed algorithm, we also build a cloud computing testbed. Real workload trace-driven simulations and testbed-based experiments show that, our algorithm greatly shortens the average finish time of map-reduce tasks and reduced time delay of web applications. Simulation results showed that our algorithm considerably reduced the amount of high-delay jobs, lowered the average traffic passed through high-level switches and improved the communication ability among virtual machines.

Keywords: Data Center, Network-Aware, Virtual Machine Re-Scheduling.

1 Introduction

With the development of virtualization technology and promotion of the concept of cloud computing, a growing number of users, especially small and medium enterprise users choose to lease computing resources from large cloud computing providers like Amazon and Google instead of building their own computing facilities. Increasing demand for cloud computing resources poses a challenge to efficiently managing users' request and physical resources.

Traditional researches usually focused on improving resources such as CPU and memory utilization by introducing intelligent virtual machine placement algorithms [2][3] and periodically virtual machine consolidation [4]. These studies often turned the problem into multi-dimensional classical bin-packing problem which is known to be NP-hard and solved by heuristic algorithms. Recently, more studies paid their attention to improving communication ability of virtual machines since numerous of distributed computing tasks like map-reduce have been deployed into datacenters. Such distributed computing tasks involve massive data transfer among VMs which require high network bandwidth guarantee.

Since distributed tasks require high bandwidth to guarantee their performance, cloud managers have to assign VMs with large mutual bandwidth usage to host machines in close proximity. However, though a variety of network-aware virtual machine placement algorithms have been proposed, at the best of our knowledge, none of these researches considered a specific network-aware re-scheduling algorithm. Caused by finish of jobs or expiration of lease, related virtual machines would be shut down and leave the system which generate plenty of resource fragments. Such fragments lead to low resource utilization as well as increased network delay. Therefore, we designed a self-adaptive network-aware virtual machine re-scheduling algorithm which automatically detects resource fragments and high cost network communication VMs and then re-schedules them through appropriate live migrations. The target of our algorithm is to maximize communication ability among VMs to improve their performance with relatively low cost.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 introduces the datacenters' architecture and analyzes the network influence on typical application in datacenters. Section 4 formulates the virtual machine placement and re-scheduling problem and proves the complexity. Section 5 explains the proposed self-adaptive network-aware virtual machine re-scheduling algorithm. After that, Section 6 presents the experimental results in simulation and in our cloud computing testbed and Section 7 gives the conclusion of this paper.

2 Related Work

Meng Wang et al. [1] formulated the virtual machine consolidation into a Stochastic Bin Packing problem and proposed an online packing algorithm. *Zhu Jiang* et al. [17] studied the problem of virtual machine allocation under the consideration of providing bandwidth guarantees and proposed an online allocation algorithm for tenants with homogeneous bandwidth demand, which aimed to improve the accuracy of existing algorithms. *Meng* et al. [5] proposed a traffic-aware virtual machine placement algorithm to improve the network scalability which tried to allocate VMs with large mutual bandwidth usage to host machines in close proximity. *Alicherry* et al. [6] extends the network-aware virtual machine placement problem into distributed Cloud. They divided the problem into two stage, firstly choosing datacenter and then choosing rack and server and proposed a 2-approximation algorithm. And *Jiang* et al. [7] study a joint tenant (e.g., server or virtual machine) placement and routing problem and gave an approximation to minimize traffic costs. *Ofer Biran* et al. [9] proposed a heuristics Network-Aware VM Placement algorithm which trying to allocate a placement that not only satisfied the predicted communication demand but was also resilient to demand time-variations. *Wilson* et al. [14] studied influence of bandwidth on web applications in datacenter and proposed a deadline-aware control protocol to lower flow latency and improve burst tolerance.

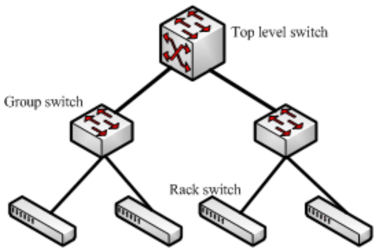
Breitgand et al. [16]studied the cost of reconfiguring virtual machines in response to workload variations. they observed that live migration requires a significant amount of spare CPU on the source server and if spare CPU is not available, it impacts both the duration of migration and the performance of the application being migrated. *Alexander* et al. [12] introduced network topology aware scheduling models which took workload characteristics, network bandwidth requirements of migrations and network topologies into account. *Shrivastava* et al. [10] studied the inherent dependencies between VMs and the complex load interactions between the underlying physical server. They introduced an application-aware virtual machine migration algorithm which incorporated inter-VM dependencies and the underlying network topology into virtual machine migration decisions.

3 Datacenter Architecture and Typical Application

The most typical network structure in datacenter is a tree or tree-like topology which can be described in Fig.1(a). Fig.1(b) is our cloud computing testbed. In datecenter, each physical server directly connects to one rack switch and each rack switch connects to a group switch (aggregate switch). Finally, each group switch connects to top level switch. In the tree structure, the bandwidth of higher level switch is shared by lower level switch which leads to decrease of bandwidth in inter-group communications. However, two typical type of applications in datacenters, distributed computing and 3-tier web application, both involves large data transfers which apparently requires high network bandwidth guarantee. Therefore, in this section, we researched on the communication influence on these two applications. We use phrase *tree-level* as the meaning of the level of network tree which traffic passed through.

3.1 Distributed Computing Scenario

Leasing datacenter's resources to run temporary distributed computing task is one of most popular applications in cloud computing and map-reduce is a typical



(a) Abstract Network Architecture

(b) Experimental Cloud Computing Testbed

Fig. 1. Network Architecture

type of distributed computing. Therefore, we choose PUMA [1], a map-reduce Benchmark, to test the network influence on the performance of MapReduce Jobs. We did our experiment under two situations, network traffic normal and overload. We compared job's finish time in different ways of virtual machine placement under these two situations.

As left part of Table 1 shows, MapReduce job's average finish time prolongs with tree level rises. And when traffic is overloaded, the negative influence on the performance of MapReduce jobs becomes more conspicuous.

Table 1. Network Influence on Tasks' Performance

Finish Time	Tree Level		
	1	2	3
Normal	1145	1262	1457
Overload	1174	1337	1807

Response Time(ms)	Tree Level		
	1	2	3
Normal	45	46	48
Overload	53	64	92

3.2 Web Application Scenario

Current Web Applications normally use 3-tier architecture. Considering the performance, the manageability and the robustness, developers would choose to deploy each tier into different physical machines or virtual machines. Therefore, the communication ability between VMs affects the overall performance of Web Application. RUBiS [2] is an auction site prototype modeled after eBay.com which is used by us to evaluate the performance of application servers (virtual machines).

As right part of Table 1 shows, in normal situation, average response time almost stay same when tree level rises. However, when traffic is overloaded, average response time increases quickly with tree level rises. The response time in worst case nearly doubled compared with the best case.

[1] Purdue MapReduce Benchmarks Suite
`http://web.ics.purdue.edu/~fahmad/benchmarks.htm`

[2] Rice University Bidding System, `http://rubis.ow2.org/`

4 Virtual Machine Placement and Re-Scheduling

In the previous section, we found that the tree level where inter virtual machine traffic passed through have a strong impact on the performance of distributed computing and web application tasks. In addition, after a long time running, some virtual machines would be shutdown or deactivated because of the job on them has finished which may lead to resource fragments. These fragments will lead to *scattered task* phenomenon (eg. a task's virtual machines have been allocated into different racks even different rack groups). As Fig.2 shows, as time goes on, the amount of *scattered task* rises.

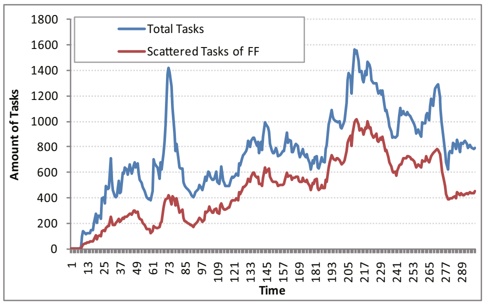


Fig. 2. Amount of *Scattered Tasks*

4.1 Virtual Machine Placement

To decrease the amount of *scattered tasks*, firstly, we give the formal definition of the virtual machine placement problem.

Suppose a datacenter has k_{RG} rack groups, each rack group has k_{R_i} servers. The total amount of physical servers is k_P. We denote each server as p_i and the physical resource which belongs it as a vector $\mathbf{H}_i$. If p_i is running, then $S(i) = 1$, otherwise $S(i) = 0$. $RG(i) = i_{RG}$ and $R(i) = i_R$ means that server p_i belongs to i_{RG}th rack group and i_Rth rack in this group. h_{ij} is the min tree level between p_i and p_j which could be computed by $RG(i)$ and $R(i)$, and we define $h_{ii} = 0$.

Consider the situation that a user u_k initiate a task and this task request for a group of VMs G_k and the amount of G_k is w_k. Each virtual machine v_i^k in the G_k has its specific requirement for CPU cycles and memory sizes which denotes as vector $\mathbf{R}_i^k$, and $\mathbf{R}_i^k$ could be extended to include other resources such as I/O operations and outlet bandwidth according to specific application. T_k is the $w_k \times w_k$ traffic matric of G_k and each item t_{ij}^k in T_k is the traffic between v_i^k and v_j^k during time Δt. The traffic matric could either be given by user or be calculated by service provider through the collected data. We denote $Z_i^k(m) = 1$ if virtual machine v_i^k is hosted on a physical server p_m, otherwise $Z_i^k(m) = 0$. Therefore, a feasible decision space for virtual machine placement is characterized by

$$\mathcal{Z} = \left\{ \left\{ Z_i^k(m) | Z_i^k(m) \in \{0,1\}, \sum_{m=1} Z_i^k(m) = 1, \ \forall (i,k); \right. \right. \\ \left. \left. \sum_{k=1} \sum_{i=1}^{w_k} Z_i^k(m) \cdot \mathbf{R}_i^k \leq \mathbf{H}_i \cdot S(m), \ \forall m \right\} \right\} \quad (1)$$

where the first equation guarantees that each virtual machine is placed on exactly one host, and the second equation ensures that the total resource consumptions of virtual machines on a physical server should not exceed their host's physical capacity in the condition that the machine is turned on. $HV_Z(v_i^k, v_j^k)$ is the min tree level between v_i^k and v_j^k in the placement Z.

$$HV_Z(v_i^k, v_j^k) \ = \ \sum_{m=1} \sum_{n=1} \left[h_{mn} \cdot Z_i^k(m) \cdot Z_j^k(n) \right] \quad (2)$$

4.2 Traffic Pattern Modeling

Since we have the specific position of each virtual machine and the traffic matrix between virtual machines, we could calculate the total traffics of each switch and then judge whether it is overloaded. We mark all switches in the datacenter as $\{s_0, s_1 \cdots\}$. $L(m,n) = (b_0, b_1, b_2 \cdots), b_i \in \{0,1\}$ is a route function between p_m and p_n, $b_i = 1$ means traffic between p_m and p_n would pass through switch s_i, otherwise traffic would not pass through switch s_i. After we defined $L(m,n)$ between p_m and p_n, we could define $LM(v_i^k, v_j^k)$ as the route function between v_i^k and v_j^k.

$$LM(v_i^k, v_j^k) \ = \ \sum_{m=1} \sum_{n=1} \left[L(m,n) \cdot Z_i^k(m) \cdot Z_j^k(n) \right] \quad (3)$$

Therefore the total traffic of a task on all switches could be calculated as follow.

$$D(k) \ = \ \sum_{i=1}^{w_k} \sum_{j=1}^{w_k} \left[LM(v_i^k, v_j^k) \cdot t_{ij}^k \right] \quad (4)$$

4.3 Measurement of Network Benefit

As showed in previous sections, an overloaded switch has a strong negative effect on the performance of virtual machines which have large traffic passing through it. Since we have already got all tasks' traffic and their route, we could get the total traffic of switch s_i,

$$DS(i) \ = \ \sum_{k=1} D(k) \cdot \mathbf{1}_i \quad (5)$$

where $\mathbf{1}_i$ means a vector where only ith bit equals 1, other bits equal 0. If $DS(i)/\Delta\ t \geq C(i)$, $C(i)$ is capacity of switch s_i, s_i is overloaded. However, it is obviously too late to re-schedule while a switch is overloaded. Therefore, we introduce parameter α, if $DS(i)/\Delta\ t \geq \alpha \cdot C(i), \alpha \in (0,1]$ and s_i is not the rack switch, it will trigger re-scheduling.

Assume at time t_0, the placement of virtual machines was $\left\{X_i^k(m)\right\}$, $\left\{X_i^k(m)\right\} \in \mathcal{Z}$; and after re-scheduling, the new placement of virtual machines is $\{Y_i^k(m)\}$, $\{Y_i^k(m)\} \in \mathcal{Z}$.

Our target is to improve the communication capability among VMs in each group G_k,thus, a practical and effective way is to keep all VMs of a group in a close position(eg. schedule them into the same server or the same rack). It has three advantages. Firstly, the lower tree level traffic has to pass through, the higher bandwidth it may shared. Secondly, if traffic only pass through low tree nodes (switches), it could save the network capacity of high tree nodes which means more bandwidth could be shared by lower nodes. Thirdly, as Table 1 shows, overload occurs in lower level switches has less influence on tasks compared with overload occurs in higher level switches.

Therefore, we define a $Benefit$ function to estimate network influence quantitatively.

$$Benefit_Z(G_k) = \sum_{i=1}^{w_k} \sum_{j=1}^{w_k} \frac{t_{ij}^k}{B(HV_Z(v_i^k, v_j^k))} \tag{6}$$

where $B(HV_Z(v_i^k, v_j^k)$ is a bandwidth function of min tree level. To simplify the computation, we define $B(x) = (N)^x$, N is the oversubscription rate. According to Cisco's Datacenter Infrastructure Design Guide[3], a typical network design of datacenters has a oversubscription rate of 2.5:1 to 8:1. Here we use 4:1 as our oversubscription rate in experiment and it should be modified to real oversubscription rate on the basis of specific datacenter architecture.

4.4 Re-Scheduling Optimization

Suppose at time t_0, the placement of virtual machines was $\left\{X_i^k(m)\right\}$, $\left\{X_i^k(m)\right\} \in \mathcal{Z}$; and after re-scheduling, the new placement of virtual machines is $\{Y_i^k(m)\}$, $\{Y_i^k(m)\} \in \mathcal{Z}$. We can formulate our Network-Aware Re-Scheduling (NARS) problem as follows:

NARS(Y)

Maximize $\sum_{i=1}^{K} \ Benefit_Y(G_k)$

Subject To $\{Y_i^k(m)\} \in \mathcal{Z}, \ \forall i, \, k, \, m$

$Benefit_X(G_k) \leq Benefit_Y(G_k), \ \forall k$

The first equation is to optimize overall communication ability of the datacenter. The following equations guarantee that the new placement would be valid and for each group of virtual machines, it would benefit from re-scheduling.

[3] `http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf`

5 Re-Scheduling Algorithms

Since *Benefit* function is non-linear and virtual machine placement is a integer programming problem, NARS is a non-linear integer programming problem. Murty et al. [15] proved that non-linear programming is NP-Hard. Therefore, non-linear integer programming, as a sub-problems of non-linear programming, is NP-Hard either. Since current datacenters usually have more than 10,000 physical servers and 100,000 virtual machines running on them, to get the exact solution of the above NP-Hard problem in such a large scale is extremely time-consuming which is unacceptable in real applications. To solve the above optimization in a feasible time, we design a heuristic algorithm.

5.1 Network-Aware Re-Scheduling

Before introducing our algorithm, we give three important observations first.

– Since live-migration is a costly operation which will generate large burst traffic, moving all virtual machines of a group to new servers or rack is too expensive.
– Traffic between virtual machines is irregular, therefore, virtual machine pair which generates large traffic should have priority to be re-schedule.
– Communication through high level switch has higher delay and lower bandwidth. Therefore, virtual machine pair which communicates through high level switches should be re-scheduled in priority.

Due to these observations, we defined a priority function,

$$Priority(G_k) \;=\; \sum_{i=1}^{w_k}\sum_{j=1}^{w_k} t_{ij}^{k} \cdot [B(HV_X(v_i^k, v_j^k))] \qquad (7)$$

The value of $Priority(G_k)$ reflects the order of urgency. Thus, our algorithm firstly calculate *Priority* value of each virtual machine group and sort them in descending order. Then we select virtual machine group to do our re-scheduling in order. While processing each virtual machine group, we trying to move virtual machine pair which has large mutual traffic into the same rack with minimum operations. The detailed algorithm is described as follows.

To avoid burst traffic from live migration which may congest the network, we keep the migration speed under v_i which satisfies $DS(i)/\varDelta\, t + v_i \cdot L(j,k) \le C(i)$.

5.2 Self-Adaptive Mechanism

We adopt two different ways to execute our re-scheduling algorithm, regularly and self-adaptively. Regular way means we do our re-scheduling algorithm periodically. Short period achieves better performance but brings more cost. Large period is more efficient but may miss the best time to do re-scheduling. Self-adaptive method resorts to a more intelligent way. It checks switches' current

Algorithm 1. Network-Aware Greedy Re-Scheduling

```
Input: $\{\mathbf{X_i^k(m)}\}$
 1: Calculate each VM group's *priority* value, $\{(G_0, pri_0), (G_1, pri_1), \cdots (G_k, pri_k)\}$
 2: Sort by *priority* value in descending order, process VM group from beginning.
 3: for Each VM group $G_i$ do
 4:    Partition $G_i$ into small parts $\{SG_0, SG_1, \cdots\}$ according to which rack they belong to.
 5:    For each two parts in $\{SG_0, SG_1, \cdots\}$ , calculate their total traffic $tp_{jk}$.
 6:    Sort $tp_{jk}$ in descending order.
 7:    for Each $tp_{jk}$ do
 8:       if VM amount of $SG_j \leq SG_k$ then
 9:          Get Rack Info $Rack_k$ of $SG_k$
10:          if $Rack_k$ have enough space to
             accommodate VMs in $SG_j$ then
11:             Migrate VMs in $SG_j$ into $Rack_k$, Update $\{Y_i^k(m)\}$.
12:          end if
13:       end if
14:    end for
15: end for
16: return $\{\mathbf{Y_i^k(m)}\}$;
```

status and traffic history then judges whether re-scheduling is necessary. Self-Adaptive Re-Scheduling algorithm described as follow.

As **Self-Adaptive Trigger** described, we calculated the total traffic of each switch and compared it with their capacity. If the total traffic approaches the capacity, it denotes that the switch is overloaded or will be overloaded in near future. When switch is overloaded, we compare the current traffic of each virtual machine group with their history data. If current traffic conspicuously declines, this virtual machine group may have a strong possibility of suffering insufficient bandwidth. In such condition, re-scheduling is necessary.

Algorithm 2. Self-Adaptive Trigger

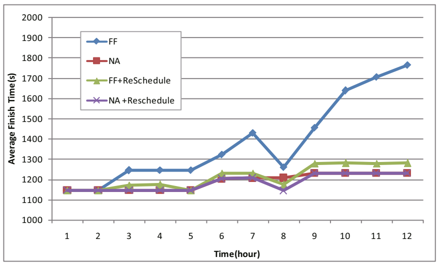
```
Input: Each Switch's Capacity $\{\mathbf{C_0, C_1}..\}$
 1: Initiate each VM group's traffic to $T_k(t_0) = \mathbf{0}$,
 2: for Each $\Delta t$ do
 3:    Calculate average traffic of each switch $DS(i)/\Delta\ t$
 4:    Calculate each VM group's traffic $T_k(t_j)$.
 5:    if $DS(i)/\Delta\ t \geq \alpha \cdot C_i$ then
 6:       Calculate average traffic of each VM group relates to this Switch $i$ in history
 7:       if Average traffic in history $\leq$ current traffic multiply $\beta$ then
 8:          Trigger Re-Scheduling
 9:       end if
10:    end if
11: end for
```

6 Experiments

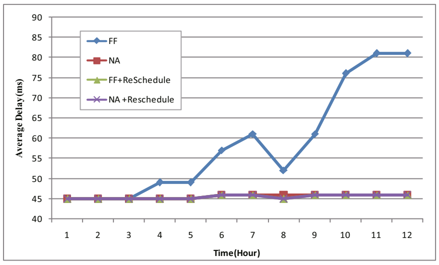
To evaluate the effectiveness of the proposed approach, we firstly conducted experiments on our experimental datacenter with small data set. After proving the effectiveness on small data set, we conducted a group of simulation experiments with large data set. Data set was based on the log of real parallel workload[4]. Small data set was derived from a small part of above data set and was modified to fit the scale of our experimental datacenter. Since this log only includes start time, end time, amount of machines, CPU requirement, Memory requirement, we have to generate a random traffic matrix for each task to fit our simulation.

6.1 Experiments in a Cloud Computing Testbed

Our experimental cloud computing testbed, see Fig.1(b), contains two rack groups and each rack group has two five-server racks. All the servers are connected by tree-like network.



(a) Effects on Distributed Computing Tasks



(b) Effects on Web Applications

Fig. 3. Experiment results in Cloud Computing Testbed

As Fig.3(a) shows, four lines represents the performance of map-reduce tasks of barely using FF (First-Fit), Network-Aware [6] and using them along with our re-scheduling algorithm respectively. Performance deteriorated quickly when barely using FF algorithm. Using our re-scheduling algorithm with FF could significantly promote the performance of FF algorithm. Due to constraint of the scale of our experimental environment, our re-scheduling algorithm didn't bring much improvement to Network-Aware allocation algorithm. However, in long-time simulation with large data set which would be shown in next subsection, our re-scheduling algorithm brought much improvement to Network-Aware allocation algorithm either.

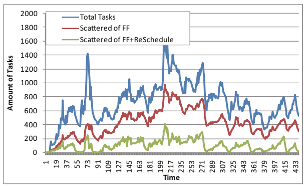
Fig.3(b) is the result of comparing the performance of Web Applications in four different algorithms. Same as map-reduce scene, delay increased quickly

[4] http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

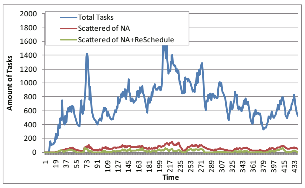
when barely using FF algorithm. And Using our re-scheduling algorithm with FF also could achieve good performance. Due to the reason mentioned in previous sub-section, our re-scheduling algorithm didn't bring much help to Network-Aware allocation algorithm in this situation.

6.2 Simulation Experiments

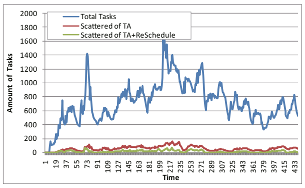
In this part, we use large data set to conduct our simulation experiments. Since it is hard and inaccurate to compute a task's finish time in simulation, we use the amount of $scatterdtasks$ and the sum of $benefit$ to indicate the performance of tasks. $Scatterdtask$ means that virtual machines which belong to this task has been allocated to different racks. If a task is scattered, its communication ability would be weaken which may lead to decline in performance. $Benefit$ function defined in equation (6), which indicates the communication ability more accurately. According to the document of Thunder Linux Cluster[5], we construct a virtual datacenter which contains 21 rack groups, 269 racks and 3224 physical servers. Workload includes 120, 000 tasks in a month.



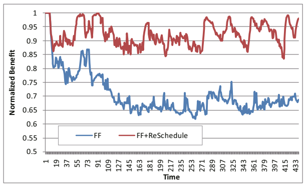
(a) *Scattered Tasks*: First-Fit with Re-Scheduling



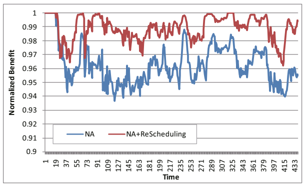
(b) *Scattered Tasks*: Network-Aware with Re-Scheduling



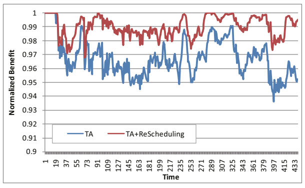
(c) *Scattered Tasks*: Traffic-Aware with Re-Scheduling



(d) Normalized $Benefit$: First-Fit with Re-Scheduling



(e) Normalized $Benefit$: Network-Aware with Re-Scheduling



(f) Normalized $Benefit$: Traffic-Aware with Re-Scheduling

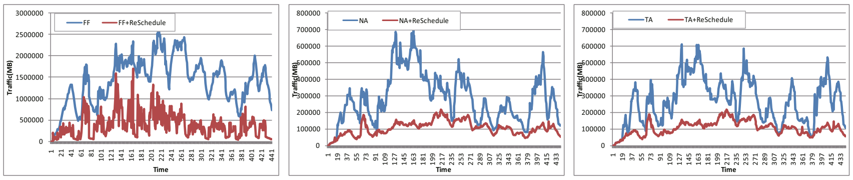
Fig. 4. Results of Simulation Experiments

Fig.4(a) shows the comparison of barely using First-Fit algorithm and First-Fit algorithm with re-scheduling.We can see that the amount of scattered tasks increase quickly with First-Fit algorithm. When we add our re-scheduling algorithm to First-Fit, the amount of scattered tasks decrease significantly. Fig.4(d) uses $benefit$ value as an indicator of network performance of tasks which had the same tendency of Fig.4(a).

[5] `https://computing.llnl.gov/?set=resources&page=index`

Fig.4(b) shows the comparison of barely using Network-Aware algorithm[6] and Network-Aware algorithm with re-scheduling. We can see that, although barely using Network-Aware VM allocation algorithm could achieve good performance in scattered tasks, it could do better when it coordinated with our re-scheduling algorithm. Fig.4(e) uses $benefit$ value as an indicator of network performance of tasks. It also shows that our re-scheduling algorithm could promote performance of original Network-Aware algorithm.

Fig.4(c) shows the comparison of barely using Traffic-Aware algorithm[5] and Traffic-Aware algorithm with re-scheduling. We can see that, Traffic-Aware algorithm achieved good performance as Network-Aware algorithm. However, it also achieved better performance when it coordinated with our re-scheduling algorithm. Similar as previous results, Fig.4(f) uses $benefit$ value as an indicator of network performance of tasks and shows that our re-scheduling algorithm could promote performance of original Traffic-Aware algorithm.

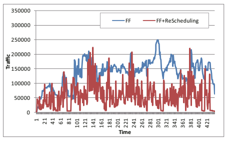


(a) First-Fit With Re-Schedule (b) Network-Aware With Re-Schedule (c) Traffic-Aware With Re-Schedule

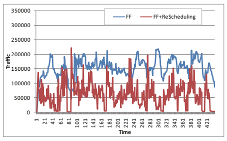
Fig. 5. Top Level Switch Traffic

Fig.5 shows the average traffic passed through top-level switches. We attached our re-scheduling algorithm with First-Fit allocation, Network-Aware allocation algorithm and Traffic-Aware allocation algorithm. Results shows that in all situations, using allocation algorithm with our re-scheduling algorithm could reduce top-level switch traffic significantly. Compared with First-Fit, Network-Aware allocation and Traffic-Aware allocation had much less top-level switch traffic. However, Network-Aware and Traffic-Aware allocation in company with our re-scheduling algorithm can make top-level switch traffic decreased further more.

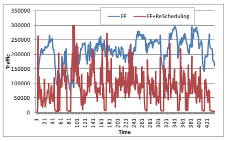
Fig.6 shows the average traffic passed through middle-level switches. Due to space constraints, we only displayed the traffic of first four groups. And since the differentiation of traffic between Traffic-Aware virtual machine algorithm and Network-Aware algorithm is not significant, we only displayed the result of Network-Aware algorithm. Similar with top-level switch traffic, results show that no matter we use First-Fit or Network-Aware allocation algorithm, when we use our re-scheduling algorithm to assist them, the middle-level switch traffic would be reduced tremendously. And since live migration renders large traffic in short time, there were some leaps in Fig.6(e),Fig.6(f),Fig.6(g),Fig.6(h) . And because the total traffic was low, these traffic leaps will not cause side effects.



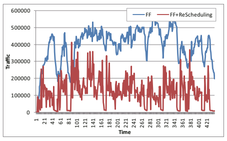
(a) Group0



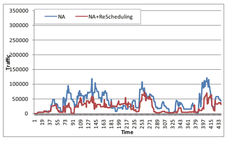
(b) Group1



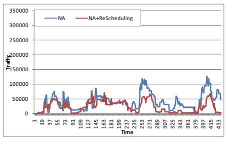
(c) Group2



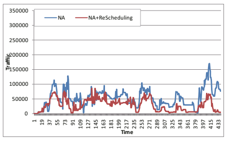
(d) Group3



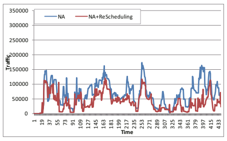
(e) Group0



(f) Group1



(g) Group2



(h) Group3

Fig. 6. Middle Level Switch Traffic

7 Conclusion

Virtual machine allocation and scheduling are key issues of management in datacenters. In this paper, we analyzed the network influence on distributed computing tasks and web applications and proposed a network-aware virtual machine re-scheduling algorithm with self-adaptive mechanism. Our algorithm focused on lowering communication cost among virtual machines through conditional and automatic virtual machine live migrations. We used two benchmarks, RUBiS and PUMA, in our experimental cloud computing testbed to examine network influence on tasks. And by using real workload data both in simulation and our experimental testbed, we compared the result of with and without using our algorithm with different virtual machine allocation algorithms. Results in real experiments showed that our algorithm reduced the average finish time of map-reduced tasks and reduce time delay of web applications. Simulation results showed that our algorithm considerably reducing the amount of high-delay jobs, decreased the average traffic of high-level switches and improving the communication ability among virtual machines.

Acknowledgments. The authors want to thank Weicheng Huai, Zhigang Jiang and Kaiyuan Wen for their novel suggestions and kind assistance. This work is partially supported by the National Natural Science Foundation of China under Grant No. 61202113, 61073028, 61021062; Jiangsu Natural Science Foundation under Grant No. BK2011510, BE2013116 and HUAWEI Project, No. YBIN2011056; JSPS KAKENHI Grant Number 25880002, 26730056 and JSPS A3 Foresight Program.

References

1. Wang, M., Meng, X., Zhang, L.: Consolidating virtual machines with dynamic bandwidth demand in data centers. In: Proceedings of International Conference on Computer Communications. IEEE (2011)

2. Van Nguyen, H., Dang Tran, F., Menaud, J.M.: Autonomic virtual resource management for service hosting platforms. In: Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE (2009)
3. Xu, J., Fortes, J.A.: Multi-objective virtual machine placement in virtualized data center environments. In: Green Computing and Communications, IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing. IEEE (2010)
4. Dutta, S., Verma, A.: Service deactivation aware placement and defragmentation in enterprise clouds. In: Proceedings of the 7th International Conference on Network and Services Management. IFIP (2011)
5. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of International Conference on Computer Communications. IEEE (2010)
6. Alicherry, M., Lakshman, T.V.: Network aware resource allocation in distributed clouds. In: Proceedings of International Conference on Computer Communications. IEEE (2012)
7. Jiang, J.W., Lan, T., Ha, S., Chen, M., Chiang, M.: Joint VM placement and routing for data center traffic engineering. In: Proceedings of International Conference on Computer Communications. IEEE (2012)
8. Kliazovich, D., Bouvry, P., Khan, S.U.: DENS: Data center energy-efficient network-aware scheduling. Journal of Cluster computing 16(1), 65–75 (2013)
9. Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A., Raz, D., Silvera, E.: A stable network-aware vm placement for cloud systems. In: Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE Computer Society (2012)
10. Shrivastava, V., Zerfos, P., Lee, K.W., Jamjoom, H., Liu, Y.H., Banerjee, S.: Application-aware virtual machine migration in data centers. In: Proceedings of International Conference on Computer Communications. IEEE (2011)
11. Steiner, M., Gaglianello, B.G., Gurbani, V., Hilt, V., Roome, W.D., Scharf, M., Voith, T.: Network-aware service placement in a distributed cloud environment. Journal of ACM SIGCOMM Computer Communication Review 42(4), 73–74 (2012)
12. Stage, A., Setzer, T.: Network-aware migration control and scheduling of differentiated virtual machine workloads. In: Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society (2009)
13. Verma, A., Kumar, G., Koller, R.: The Cost of Reconfiguration in a Cloud. In: Proceedings of the 11th International Middleware Conference (2010)
14. Wilson, C., et al.: Better never than late: Meeting deadlines in datacenter networks. Journal of ACM SIGCOMM Computer Communication Review 41(4) (2011)
15. Wilson, C., Ballani, H., Karagiannis, T., Rowtron, A.: Some NP-complete problems in quadratic and nonlinear programming. Mathematical Programming 39(2), 117–129 (1987)
16. Breitgand, D., Kutiel, G., Raz, D.: Cost-aware live migration of services in the cloud. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference. ACM (2010)
17. Zhu, J., Li, D., Wu, J., Liu, H., Zhang, Y., Zhang, J.: Towards bandwidth guarantee in multi-tenancy cloud computing networks. In: Proceedings of 20th IEEE International Conference on Network Protocols. IEEE (2012)
18. Benson, T., Anand, A., Akella, A., Zhang, M.: Understanding data center traffic characteristics. Journal of ACM SIGCOMM Computer Communication Review 40(1), 92–99 (2010)

19. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The nature of data center traffic: Measurements and analysis. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference. ACM (2009)
20. Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing sla violations. In: Proceedings of 10th IFIP/IEEE International Symposium on Integrated Network Management. IEEE (2007)
21. Jung, G., Joshi, K.R., Hiltunen, M.A., Schlichting, R.D., Pu, C.: A cost-sensitive adaptation engine for server consolidation of multitier applications. In: Bacon, J.M., Cooper, B.F. (eds.) Middleware 2009. LNCS, vol. 5896, pp. 163–183. Springer, Heidelberg (2009)

A Novel Petri-Net Based Resource Constrained Multi-project Scheduling Method

Wenbin Hu* and Huan Wang

School of Computer, Wuhan University, Hubei Province, China
hwb@whu.edu.cn, 105089486@qq.com

Abstract. This work proposes an extended Petri-Nets for resource constrained project scheduling problem (RCPSP). Several important issues about Petri-Nets on RCPSP are discussed in this paper. Firstly, the most important elements in Petri-Nets on RCPSP, including waiting place, activity place, resource place and final place, are designed. At the same time, three kinds of transitions are designed. They are coordination transition, resource allocating transition and resource releasing transition. Secondly, the Petri-Nets are improved according to the varying with increase or decrease of the number of resource acquisition. In this extended Petri-net, for the foremost difference with conventional Petri-Nets is to add a place and replace a releasing transition by the delaying transition and the firing rules of transition. Extensive experiments are performed to evaluate the performances of the proposed Petri-Nets against the state-of-art algorithms. The simulation experimental results of difference sides showed that extended Petri-Nets provides a more efficient way to solve RCPSP and produces competitive results compared with other methods investigated in this works.

Keywords: RCPSP, Petri-Nets, Simulation, modeling.

1 Introduction

RCPSP is widespread in engineering fields, such as job shop scheduling, flexible manufacturing systems and so on. In real engineering projects, there are usually multiple projects in parallel and competition for resources among projects, which increase the complexity of project scheduling. The resources allocation is necessary to consider the project itself and among projects. In addition, there is competition among the executed duration of projects, which leads to use some scheduling methods of the single project problem to the multi-project very difficultly. So how to model and solve multi-project in the environment of resource constrained is the key point of this research field. This paper will focus on the modeling technology and do some research on the RCPSP solution performance.

In order to minimize the maximum completion time of the RCPSP, the approach based on the integer programming (IP) with Lagrange multiplier has attracted great attention where some multi-constraints in RCPSP. Although IP is an effective

* Corresponding author.

method, the RCPSP formulation is not easily made for feasible solution, since RCPSP is inherently the IP function with dynamically changing resources constraint. So many constraint properties have been ignored in this method. Meta heuristics such as Genetic Algorithm [1], Tabu Search [2], Particle Swarm Optimization [3] and Bee Swarm Optimization [4] have also been used for the RCPSP. Although all of these algorithms have been easily formulated, they often leaded to infeasible answer or not good solutions, taking generally very long optimizing search time. These would be more remarkable when the scale of RCPSP becomes more and more big and more and more constrains are taken into consideration. The heuristics algorithms have defects to solve these problems. Artificial intelligent algorithms can often generate a feasible solution, and sometimes it is difficult to know whether there is a better solution.

Petri-Net is an ideal modeling tool to establish system constraints, to provide system information by mathematical analysis and provide guide line for the scheduling process. In recent years, Petri-Net was used to model and analyze various types of scheduling problems and had achieved good results. Gradisar [5] classified scheduling rules and established colored Petri-Nets Model for simulation of different scheduling rules. Ramirez [6] presented interpreted Petri-Nets to model the events and states of RCPSP system behaviour. It proposed a scheme utilizing a solution of a programming problem for RCPSP based on the interpreted Petri-Nets derived from an on-line methodology. Note that all papers in this topic focused on the discrete events modelling technology of RCPSP, but not considering the resources constraint and uncertainty duration. A series of papers have been recently presented that are based on the assumption that the duration is certainty.

From the summary of current research, many kinds of Petri-Net models were used to solve RCPSP. They focused on some key technologies of RCPSP and got better results. How to simulate the actual RCPSP system clearly is the most important goal for all researchers. There are many uncertainty duration PCPSP in actual engineering project. Some researchers used fuzzy [7], probability [6] technologies to transform uncertainty duration problem into certainty duration problem. But all of them couldn't express the detail of the states changing of all processes. In this paper, we use appropriate methods to extend Petri-Net that is able to simulate the actual system clearly according to the characteristics of practical problems. Because of the characteristics of certain duration and variable duration in RCMPSP, we take the extended Petri-Net to model certain duration problem, which divide token into logical token and resource token, and token changing represents the execution of processes and allocation of resources. Through the classification of places and transitions and adding time to places, the places are divided into activity place (AP), resource place (RP), waiting place (WP) and final place (FP). The transitions contain coordination transition (CT), resource dispatching transition (RDT) and resource releasing transition (RRT). WP and AP reflect the timing relationship among tasks by taking a balance of transaction connection. The resources competition is reflected by taking up and release for resources among tasks by RP, RDT and RRT. According to the uncertainty problem's characteristics, we add the place and transitions type, modifying rules of transition of triggering to the extended Petri-Nets. The novel strategy is presented in the extended Petri-Nets to describe the actual system of uncertainty duration problem. Through the investigation of the transition between Petri-Net models with the EM-Plant simulation models, we primarily associate the three basic elements of Petri net including name, properties and behavior.

2 Modelling of RCPSP

2.1 Ready for Modelling

2.1.1 Building of the Project Network Diagram

The logical structure of projects can be determined by analyzing the project tasks. We take virtual nodes to combine multi-projects, thus form the project network diagram. It supposes that there are P projects, which are numbered i (i =1, 2, 3, ..., P), the task number of project i is Ni, in which project process is numbered (i = 1, 2, 3, ..., Ni). Set two virtual task nodes $Tstart$ and $Tend$, which take no time to implement and don't need resources. $Tstart$ represents the beginning of all tasks, and $Tend$ represents the ending of all tasks. So we can put multiple projects into a large network diagram by setting the virtual task nodes. The multi-projects network diagram is shown in Fig.1.

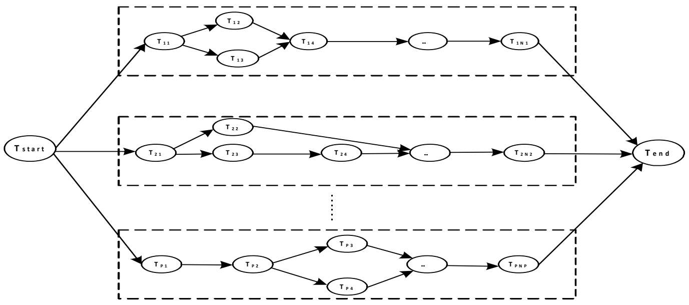


Fig. 1. Multi-projects network diagram

2.1.2 Scheduling Process of Multi-projects

There are 4 task sets in the schedule of multi-projects, which are (1) Finishing task set (Fs); (2) Going task set (Gs); (3) Waiting task set (Ws) (They are the collection of tasks to satisfy the timing constraints but not satisfy the resource needs.); (4) Non-executable task set (Ds). The task sets are shown in Fig.2.

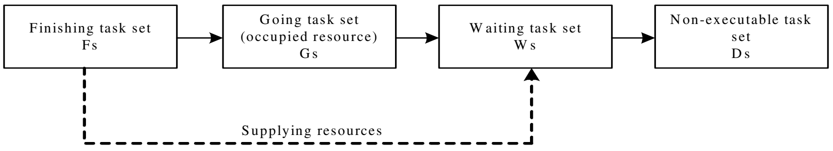


Fig. 2. Task sets of multi-project scheduling processes

2.2 Certainty Duration Problem Modelling

2.2.1 Problem Description and Modelling Assumptions

For RCPSP with certain duration, it can be described: there are R kinds of shared resources for the projects. It assumed that there are P projects. The implementation of task is based on occupying one resource. The daily supply of all resources is limited. The competition for resources is not only reflected among tasks in projects, but also reflected between different projects. So the goal of management is: how to allocate these shared resources to different tasks in projects by effective resource scheduling in order to complete all these projects in a shortest makespan.

Before mathematical modelling, it is necessary to make a reasonable simplification for the complex practical problems. For the research convenience, we set several model assumptions as following.

(1) Projects are not independent, which means there is no predecessor relationship among projects.

(2) In each project only one shared resource is needed to execute the task.

(3) The execution time of each task is determined when the resource supply is satisfied.

(4) All shared resources are renewable resources that mean the occupied resources will be released after the ending of task.

(5) The daily supply for each shared resource is fixed, including the type and quantity.

(6) Executing task cannot be interrupted.

2.2.2 Mathematical Description of Model

Symbol definitions in the description:

X represents project, X_i represents the i-th project, the number of projects is P, M_i represents the total number of items in the i-th project.

j represents task number in each project.

d_{ij} represents duration of j-th task in i-th project.

T_{iM_i} represents completed time of the last task in i-th project.

B_{ij} represents starting time of j-th task in i-th project. E_{ij} represents ending time of j-th task in i-th project.

F_{ij} represents predecessor task set of j-th task in i-th project.

t represents time serial number, $G(t)$ represents execution status of tasks set in time t.

k represents resource serial number, K represents the total number of resources required by projects.

R_k represents k-th resource daily supply.

r_{ijk} represents number of k-th resource required by j-th task in i-th project.

The mathematical representation of decision-making goal is that the total duration to execute projects P is shortest:

$$\min\left\{\max_{i=1}^{P}\left\{T_{iM_i}\right\}\right\} \tag{1}$$

Mathematical representations of the decision-making constraints are as follow.

(1) Timing constraints between the internal task executions of project:

$$B_{ih} + d_{ih} \leq B_{ij}, \forall h \in F_{ij}, \forall i, j \tag{2}$$

(2) Resource constraints, the demand for resources can't be greater than the supply of resources in every moment:

$$\sum_{i=1}^{P}\sum_{j=1}^{M_i} r_{ijk} \leq R_k, \forall i, j \in G(t) \tag{3}$$

2.2.3 Petri-Net Modelling

According to the characteristics of RCPSP with certainty duration, this paper constructs extended Petri-Nets with the object of multi-project tasks set. The extended Petri-Nets can be described by 9-tuple.

$$EXPN = \langle P, T, TYPEP, TYPET, A, W, ET, resourceAmount, S_0 \rangle$$

$P = \{P_1, P_2, ..., P_m\}$ represents limited places set, $T = \{T_1, T_2, ..., T_n\}$ represents limited transition set. $TYPEP$ and $TYPET$ represent type of place and type of transition. We mainly construct three types of places and types of transition. The place type and transition type are shown in Fig.3 and Fig.4.

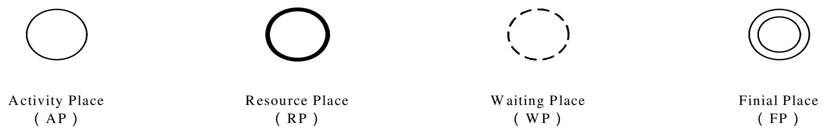


Fig. 3. Four kind of places

One activity of project is represented by two places. One is named waiting place (WP), the other is named activity place (AP). WP is front of AP. $EXPN$ is also defined by two token types, which called logical token (LT) and resource token (RT). LT in AP represents that task is running. LT in WP represents that task is waiting resources. The resources will be diverted to AP once they occupy the resources.

In addition, a resource pool (RP) represents a type of resource. The RP token is a resource token. It represents the number of available resources. Final place (FP) represents ending of project. FP has a logical token when all of projects have ended. In addition to RP, each place only belongs to one of the projects.

Coordination transition (CT) is used to connect to WP, AP and FP. CT is equivalent to the process of coordinating project running. RDT is used to connect the input RP, WP, output RP and schedule resources. Resource release transition (RRT) is used to connect AP, RP and recycle resources.

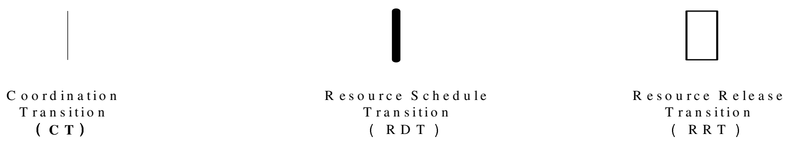

Coordination Transition (CT)
Resource Schedule Transition (RDT)
Resource Release Transition (RRT)

Fig. 4. Three kinds of transition

A represents the collection of arcs between places and transitions. W represents the collection of weight on these arcs and the number of tokens I/0, which is set to default 1 for simplify. When weight is 1 it is not impossible to mark weight on arcs.

resourceAmount represents the number of resources in RP that is the number of resource tokens.

S_0 represents the initial status of $EXPN$, which is described by three-tuples $\langle M_0, AET_0, RA_0 \rangle$, M represents places' (in addition to RP) token status. In addition to RP, other places can only have one token (logical token), so the token status of each place can be represented by {0, 1}. AET represents projects' continued time. AET represents the time spent by all projects implementation when the token status of FP is 1. RA represents the number of resources in RP.

In RCPSP, there is always competition for some resources among projects. The relationship is shown in Fig.5.

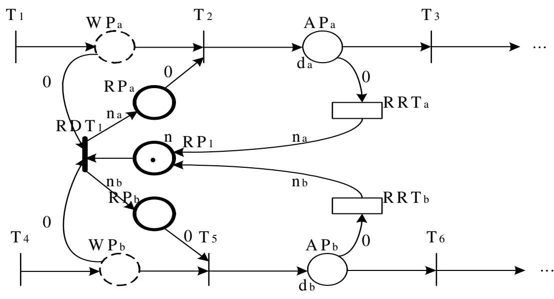


Fig. 5. Resource competition of certainty duration problem

Fig.5 shows that the execution of task a and task b need to compete for resource RP_1. Task a needs n_a resources. Task b needs n_b resources. The number of RP_1 is n. When WP_a and WP_b get tokens (logical token), it represents that task a and task b satisfy the predecessor relationship. When resources are satisfied, tokens of WP_a or WP_b will immediately be transferred to AP_a and AP_b. There are four situations as following.

(1) $n \geq n_a + n_b$. It represents that resources is enough and can satisfy task a and task b. when RDT_1 is triggered, n_a tokens and n_b tokens of RP were transferred to RP_a and RP_b. Once RP_a and RP_b have tokens, T_2 and T_5 is triggered. WP_a and WP_b is transferred to AP_a and AP_b.

(2) $n \geq n_a$ but $n < n_b$; or $n \geq n_b$ but $n < n_a$. It represents only task a or task b is satisfied. It assumed that task a is satisfied. When RDT_1 is triggered, only n_a tokens in RP_1 is transferred to RP_a . Once RP_a has tokens, T_2 is triggered and tokens of WP_a is transferred to AP_a . Task a is implemented. On the contrary, if RP_b cannot get tokens, T_5 can't be triggered, and task b can't be implemented, so task b will be waiting resources.

(3) $n \geq n_a$ and $n \geq n_b$ but $n \leq n_a + n_b$. It represents resources can satisfy task a and task b, but can't satisfy both together. RDT_1 is triggered and takes a random strategy. Tokens in RP transferred randomly, so it is possible n_a tokens is transferred to RP_a or n_b tokens is transferred to RP_b . In other words, it is random to execute one task and the other task will wait for resources to be released.

(4) $n < n_a$ and $n < n_b$. It represents task a and b can't both be satisfied, so RDT_1 can't be triggered and task a and task b can't be implemented.

In addition, d_a and d_b in Fig.5 represent execution time of tokens in AP_a and AP_b . When AP_a tokens enter it and spend d_a time, AP_a' token is transferred to other places. RRT_a is triggered. RP_1 is added n_a tokens. These situations represent task a is completed, and resources of task a are recycled. That is the same reason for AP_b .

2.3 Uncertainty Duration Problem Modelling

2.3.1 Problem Description and Assumptions in Modelling

In the uncertainty duration problem, the task duration of projects is uncertain, which is always related to the number of shared resources occupied. The more allocated to tasks, the less task execution duration, which tend to more realistic. In other words, when the amount of resources allocated to tasks are different, the duration of tasks is different, which is referred to the uncertainty duration. It can be described as: there are R kinds of shared resources, which are supplied to all projects waiting to start. It assumes that there are P kinds of projects, in which tasks' execution depends on occupying amount of one resource. All of resources are supplied limited. This competition for resources is not only reflected between each task in project, but also reflected between different projects. Besides, task execution duration of projects is uncertain, or variable, if a project (task) gets more of resource, then other projects (tasks) will obtain fewer resources. Which project gets more resources will accelerate the completion, and others will postpone completion. Management objectives are: under the condition of this variable duration, how to allocate these shared resources to different tasks in projects and make the completion duration of these projects be shortest by effective method of resource coordinated allocation.

Compared with the certain duration, this problem adds an assumption, that is, within a certain range, the duration of tasks, which use shared resources, is inversely proportional to the number of resources.

2.3.2 Mathematical Description of Model

In addition to the definitions of certainty duration problem, the following symbol definitions are changed and added.

d_{ij} , is changed. It represents the execution time of someone task, which does not need shared resources.

H_{ijk} , is changed. It represents the daily maximum demand of k-th resource for j-th task of i-th project.

Q_{ijk} , is changed. It represents the total demand of k-th resource for j-th task of i-th project.

D_{ij} , is changed. It represents the actual execution time of j-th task of i-th project.

Mathematical representation of decision objective is also that the total duration for P projects being executed is shortest:

$$\min\left\{\max_{i=1}^{P}\left\{T_{iM_i}\right\}\right\} \tag{4}$$

Mathematical representation of decision constraint condition is the relationship among task execution time, total amount demand of someone resource and daily assigned amount of this resources:

$$D_{ij} = \begin{cases} \dfrac{Q_{ijk}}{r_{ijk}} & \text{, when task use shared resources} \\ d_{ij} & \text{, when task don't use shared resources} \end{cases} \tag{5}$$

(2) Timing constraints between the internal task executions of project:

$$B_{ih} + D_{ih} \le B_{ij}, \forall h \in F_{ij}, \forall i, j \tag{6}$$

(3) Resource constraints, the demand for resources can't be greater than the supply of resources in every moment:

$$\sum_{i=1}^{P} \sum_{j=1}^{M_i} r_{ijk} \le R_k, \forall i, j \in G(t) \tag{7}$$

(4) The daily demand of shared resources of tasks is limited:

$$0 \le r_{ijk} \le H_{ijk} \tag{8}$$

2.3.3 The Improvement on Petri Net Model

According to the characteristics of the variable project duration, it improves the extended Petri-Nets model on certainty duration. It refines the repository and divides it into Global Repository Place (GRP) and the Local Repository Place (LRP). And then, the GRP represents a type of global resources, and the inside token (resources token) indicates the number of a certain resource that is the number of the current available resource. The LRP represents the current situation about the resource occupied by a task, which need share resources, and the number of token indicates the amount of a certain resource occupied by the task execution. In addition, the local repository pool contains two properties, which are the maximum amount of a certain resource can be assigned every day when task executes and the total amount of a certain resource required by the task execution. The number of transferring to the inside token is determined by the scheduling transition of the resource connected to it. Thus, five types of the place are shown in Fig.6.

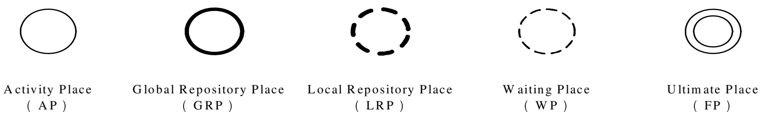


Fig. 6. Five types of the place

The classification of the transition is also improved. Three types of transition are shown in Fig.7.

In the Resource Constrained Multiple Project Scheduling problem under the environment of the variable project duration that kind of the variable project duration is reflected in the task requires sharing resources. Thus, the execution time of the task, which is not need sharing resources, is sure so that every place is given to the concept of time in the extended Petri-Nets. The ET manifests that time and is called execution time. If the ET in the Activity Place is zero, it indicates that the execution of that activity requires sharing resources and its execution time is determined by the Delay Transition connected to it. Otherwise, it indicates that the execution of that activity is not need sharing resources so the ET represents the time takes by the activity from begin to the end. In that improvement Petri net model, the competitive relation between tasks is shown in Fig.8.

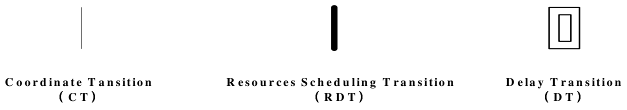


Fig. 7. Three types of the transition

Fig.8 shows four tasks a, b, c, d, where the execution of c and d does not require sharing resources and their execution time are dc and dd. On the contrary, the execution of task a and b requires the competitive resource RP1. And a new concept is proposed. It means that every task requires resources has a maximum amount of resources that it can obtain every day. When the task obtains over those resources, the task execution time is shortest.

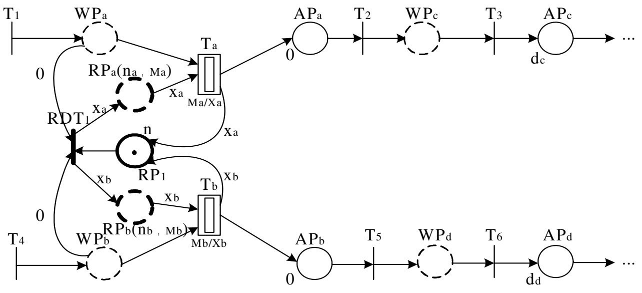


Fig. 8. the resource competitive relationship of the variable project duration

The Resource Scheduling Transition RDT_1 mainly responses the assignment of the RP_1 resource. As shown in Figure 8, that WP_a and WP_b obtain the tokens indicates that task a and b satisfy the tight conditions. At present, RDT_1 will take the allocation strategy that it randomly produces two values x_a and x_b ($0 < x_a \leq n_a$, $0 < x_b \leq n_b$) according to the attributes n_a and n_b of the Local Repository RP_a and RP_b to which it connects backward. And the attributes means the task a and b can obtain the maximum amount of the resource RP_1 every day. Then it allocates the resources according to the number of the Global Repository Place RP_1 to which it connects backward. The type of the assignment is similar to the RDT_1 shown in Fig.4 except the competitive objects x_a and x_b .

3 Numerical Experiments

The experiments simulate the Single Mode Sets Cases in PSPLIB [8] library, these cases consist of 30-120 activities. The experiments verified the certainty duration RCPSP are run for benchmarks with j30, j60, j90, and j120 case studies. The numbers of schedules are set as 1000, 5000 and 50,000. Because the durations of the cases in PSPLIB library are certainty, in order to verify the usefulness of the extended Petri-Nets of this paper, we change the certainty durations of cases j30, j60, j90 and j120, and get the new benchmarks j'30, j'60, j'90 and j'120 which lose the duration constraints. The experiments verified the uncertainty duration RCPSP are run for benchmarks with j'30, j'60, j'90 and j'120 case studies. The numbers of schedules are also set as 1000, 5000 and 50,000.

3.1 Success Rate of Solution

Ziarati [4] showed the success rate of BSO, ABC, BA, BS0-FBI, ABC-FBI and BA-FBI for cases of PSPLIB library. Table 1 shows the comparison of the proposed extended Petri-Nets of this paper with Ziarati [4] and common Petri-Nets. Table 1 shows that the proposed extended Petri-Nets of this paper have high success rates on the all problems. The performance of the proposed extended Petri-Nets increase as the complexity of the case studies increases. Table 2 shows the certainty duration problems have higher success rates than uncertainty duration problems. The performance of uncertainty durations problems decrease as the complexity of the case studies increases.

Table 1. Average success rate for all cases of PSPLIB

Solution	**Reference**	**1000**	**5000**	**50,000**
(1) the average success rate of j30				
BSO-FBI	Ziarati[4]	83.55%	90.21%	96.25%
ABC-FBI	Ziarati[4]	82.50%	90.00%	95.63%
BA-FBI	Ziarati[4]	83.96%	91.50%	97.09%
PNS		99.09%	100%	100%
Extended PNS	This study	100%	100%	100%

Table 1. (*Continued)*

Solution	Reference	1000	5000	50,000
(2) the average success rate of j60				
BSO-FBI	Ziarati[4]	72.08%	73.34%	77.09%
ABC-FBI	Ziarati[4]	71.67%	73.34%	76.67%
BA-FBI	Ziarati[4]	72.30%	73.96%	78.13%
PNS		90.24%	96.78%	96.89%
Extended PNS	This study	99.43%	100%	100%
(3) the average success rate of j90				
BSO-FBI	Ziarati[4]	72.30%	74.17%	75.42%
ABC-FBI	Ziarati[4]	71.88%	73.96%	75.00%
BA-FBI	Ziarati[4]	72.30%	74.60%	76.25%
PNS		89.12%	91.47%	92.98%
Extended PNS	This study	97.09%	98.87%	99.81%
(4) the average success rate of j120				
BSO-FBI	Ziarati[4]	29.17%	30.84%	33.00%
ABC-FBI	Ziarati[4]	29.34%	30.34%	32.67%
BA-FBI	Ziarati[4]	29.84%	31.17%	33.84%
PNS		65.78%	77.56%	81.75%
Extends PNS	This study	90.00%	93.36%	94.46%

Table 2. average success rate of extended Petri Nets for certainty duration and uncertainty duration

Cases	1000	5000	50000
j30	100%	100%	100%
j’30	97.76%	99.65%	100%
j60	99.43%	100%	100%
j’60	92.76%	96.77%	99.34%
j90	97.09%	98.87%	99.81%
j’90	88.80%	90.12%	90.67%
j120	90.00%	93.36%	94.46%
j’120	72.25%	77.89%	80.00%

3.2 Solution Speed for Certainty and Uncertainty Duration

This section discusses the solution speed of common Petri-Nets and extended Petri-Nets for PSPLIB. The solution speed is defined as the number of iterations a method requires to solve a PSPLIB instance. Fig 9 shows the solution speed of common Petri-Nets and extended Petri-Nets for certainty duration of each case in j30, j60, j90, j120 of PSPLIB. Fig10 shows the solution speed of common Petri-Nets and extended Petri-Nets for uncertainty duration of each case in j’30, j’60, j’90 and j’120.

Fig 9 and Fig 10 show the results that the extended Petri-Nets in this paper have the faster solution speed than common Petri-Net both in certainty duration and uncertainty duration of PSPLIB.

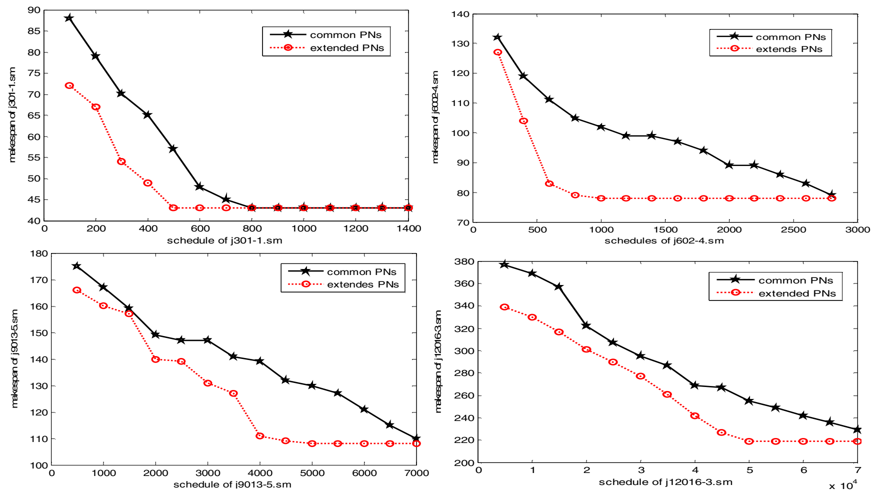


Fig. 9. Certainty duration solution behaviors of PNs and extends PNs for PSPLIB

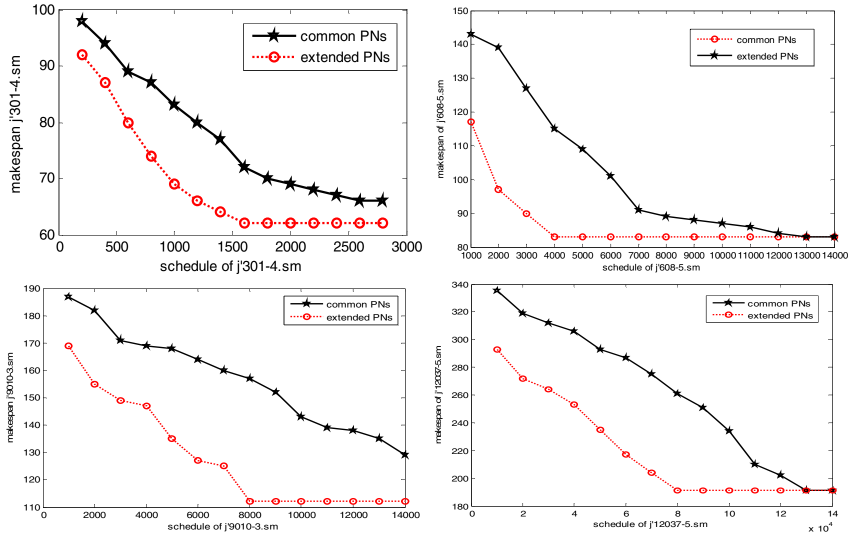


Fig. 10. Uncertainty duration solution behaviour of PNs and extends PNs

3.3 Evaluation of Adaptability for Unexpected Change of Uncertainty Duration

In order to investigate adaptability for the unexpected change of the uncertainty duration for extended Petri-Nets in this paper, we set the PSPLIB cases where only half of the total resources are committed to RCPSP after some time intervals from the beginning. We take four cases of PSPLIB (j303-5.sm, j608-9.sm, j903-5.sm, and j12010-3.sm) as testing cases. The schedules for these four cases by using extended Petri-Nets in this paper are list in Table 3. We see that there is no significant difference in the solution schedules for each case. The experiment results show that extended Petri-Nets in this paper has good performance to the unexpected changes of uncertainty duration.

Table 3. Evaluation of adaptability for unexpected changed of uncertainty duration

Unexpected changed	Schedule of solution
Case of j'303-5.sm	
All processes are executed from the beginning	1479
Half of processes are executed at 400 step	1480
Half of processes are executed at 500 step	1483
Half of processes are executed at 600 step	1489
Case of j'608-9.sm	
All processes are executed from the beginning	3960
Half of processes are executed at 1000 step	3966
Half of processes are executed at 1200 step	3990
Half of processes are executed at 1400 step	4001
Case of j'903-5.sm	
All processes are executed from the beginning	7853
Half of processes are executed at 2000 step	7910
Half of processes are executed at 3000 step	7922
Half of processes are executed at 4000 step	7958
Case of j'12010-3.sm	
All processes are executed from the beginning	60129
Half of processes are executed at 10000 step	60298
Half of processes are executed at 20000 step	60332
Half of processes are executed at 30000 step	60398

3.4 Comparative Study

In this subsection we compare the performance of extended PNs method with other approaches published in scientific literature. The objective of this experiment is to compare the performance to find the schedule with the least makespan. The experiments are conducted j60 case with 500 instances, j90 case with 600 instances. As far as this paper is concerned we have found other heuristic and meta-heuristic algorithms, including simulated annealing (SA), tabu search (TS), genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO) and bee algorithm (BA). All these methods are regarded as effective proofed and widely adopted methods to solve the RCPSP. The performances of these methods are obtained from the original literature.

Table 4-5 displays the average deviations for main methods over the j30, j60 and j120 case study. It can be observed that the extended PNs have better performance than other literature method. In general, the extend PNs method provides competitive results compared to other meta-heuristic methods investigated in this work.

Table 4. Average deviation of optimal makespan for j60 case

method	Reference	1000	5000	50000
SA-activity list	Bouleimen, [9]	12.75	11.9	---
TS	Baar etal., [10]	12.97	12.18	11.58
GA-priority rule	Hartman,[1]	13.30	12.74	12.26
ACOSS	Chen etal,. [11]	11.75	10.98	10.67
BA-FBI	Ziarati etal., [4]	12.55	12.04	11.16
Common PNs	This study	9.88	9.03	8.76
Extended PNs	This study	5.27	5.04	4.06

Table 5. Average deviation of optimal makespan for j90 case

method	Reference	1000	5000	50000
SA-activity list	Bouleimen, [9]	42.81	37.68	---
TS	Nonobe [12]	40.86	37.88	35.85
GA-activity list	Hartman,[1]	39.37	36.74	34.03
ACOSS	Chen etal,. [11]	35.19	32.48	30.56
Sampling-LFT	Kolisch [13]	42.84	41.84	40.63
BA-FBI	Ziarati etal., [4]	37.72	36.76	34.55
Common PNs	This study	39.14	38.66	38.05
Extended PNs	This study	29.19	27.06	24.09

4 Conclusions

There are many variable project duration problems in the multiple projects scheduling problem. And the different point for the sure project duration problem is that the project duration of the task execution is determined by the amount of resources assigned every day. It is not sure but ranges in an area. The project duration of the task execution is an inverse proportional relationship with the amount of the resources assigned every day. When using the Petri net model to model the problem, it mainly makes some improvement on the extent resources assignment Petri net establishing on the sure problem and it also refines the previous resources place into the Global Repository Place and the Local Repository Place added the attributes in the classification of the place. In the classification of the transition, it cancels the previous resources releasing transition, increases the delay time of the transition. The Delay Transition on one hand manifests the procedure for the task execution and on the other hand responses the resources releasing. Besides, the resources releasing transition on the way of resources assignment is different from the previous. It reflects the variable characteristics of resources assignment on variable project duration. At the same, when discussing the conversion between the Petri net model and EM-Plant model, its conversion way is similar to the sure project duration problem and is slight different from some behaviors in the conversion. In the end, the effectiveness of the method is verified by the optimal results which are obtained from the modeling simulation on the realistic examples.

References

1. Hartmann, S.: A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics 49, 433–448 (2002)
2. Verhoeven, M.G.A.: Tabu search for resource constrained scheduling. Eur. J. Oper. Res. 106, 266–276 (1998)
3. Hu, W., Liang, H., Peng, C., Du, B., Hu, Q.: A Hybrid Chaos-Particle Swarm Optimization Algorithm for the Vehicle Routing Problem with Time Window. Entropy 15, 1247–1270 (2013)
4. Ziarati, K.: On the performance of bee algorithms for resource-constrained project scheduling problem. Applied Soft Computing 11, 3720–3733 (2011)
5. Gradisar, D., Music, G.: Production-process modeling based on production-management data: A Petri-net approach. International Journal of Computer Integrated Manufacturing 20, 794–810 (2007)
6. Ramirez-Trevino, A., Ruiz-Beltran, E., Rivera-Rangel, I., Lopez-Mellado, E.: Online fault diagnosis of discrete event systems. IEEE Transactions on Automation Science and Engineering 4, 31–39 (2007)
7. Ding, Z., Bunke, H., Kipersztok, O., Schneider, M., Kandel, A.: Fuzzy timed Petri nets-analysis and implementation. Mathematical and Computer Modelling 43, 385–400 (2006)
8. Project Scheduling Problem Library- PSPLIB, `http://129.187.106.231/psplib`
9. Bouleimen, K., Lecocq, H.: A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 149, 268–281 (2003)
10. Baar, T., Brucker, P., Knust, S.: Tabu search algorithm and lower bounds for the resource-constrained project scheduling problem. In: Meta-heuristics: Advanced and Trends in Local Search Paradigms for Optimization. Klumer, Dordrecht (1998)
11. Chen, W., Shi, Y.J., Teng, H.F., Lan, X.P., Hu, L.C.: An efficient hybrid algorithm for resource-constrained project scheduling. Information Science 180, 1031–1039 (2010)
12. Nonobe, K., Ibaraki, T.: Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Essays and Surveys in Metaheuristic. Kluwer Academic Publishers, Dordrecht (2002)
13. Kolisch, R.: Serial and parallel resource-constrained project scheduling methods revisite: Theory and computation. European Journal of Operational Research 90, 320–333 (1996)

Interconnection Network Reconstruction for Fault-Tolerance of Torus-Connected VLSI Array[⋆]

Longting Zhu[1], Jigang Wu[1,⋆⋆], Guiyuan Jiang[2], and Jizhou Sun[2]

[1]School of Computer Science and Software Engineering,
Tianjin Polytechnic University, Tianjin, China 300387
[2]School of Computer Science and Technology, Tianjin University,
Tianjin, China 300072
asjgwu@gmail.com

Abstract. Effective fault-tolerant techniques are essential for improving the reliability of multiprocessor systems. This paper investigates the fault-tolerance of torus-connected VLSI array using pre-integrated spare processing elements (PEs), by reconfiguring the interconnection network among all PEs. We model the problem of whether all faulty PEs can be replaced by spare ones as the problem of finding maximum independent set for a contradiction graph, which is constructed from the original physical arrays with faulty PEs. Each node of the graph represents an alternative of a faulty PE, while an edge denotes that different alternatives cannot coexist. We propose efficient algorithms to construct contradiction graphs from physical arrays with faulty PEs and redundant PEs. We then customize an ant-colony algorithm to find independent set as large as possible. We develop an efficient algorithm to generate logic arrays based on the produced independent set. Three different distributions of redundant PEs are discussed in this paper, and satisfactory results have been achieved in simulation.

Keywords: torus-connected VLSI array, reconfiguration algorithm, fault-tolerance, contradiction graph.

1 Introduction

With the rapid development of VLSI/WSI technology, it is possible to integrate hundreds to thousands of processing elements (PEs) on a single chip. At the same time, the reliability of multi-processor system becomes an increasingly serious problem with the increasing density of VLSI. Therefore, efficient fault-tolerant techniques must be developed to improve the reliability of multiprocessor systems. It is verified that with the ever-increasing circuit density, obtaining high

⋆ This work was supported by the National Natural Science Foundation of China under Grant No. 61173032, and the Doctoral Fund of Ministry of Education of China under Grant No. 20100032110041 and No. 20131201110002.

⋆⋆ Corresponding author.

fabrication yield solely through improving the manufacturing process (for single PE) is increasingly difficult and will become unaffordable in the near future [1]. Thus, one promising method for fault-tolerance is to reorganize fault-free PEs to a new regular interconnection network, which ensures well-controlled electrical parameters, by changing the interconnection among PEs. In this paper, we investigate the fault tolerance at PE level, i.e., constructing logical VLSI arrays which contain no faulty PEs from a physical array by changing the interconnection networks among all PEs. The following two types of fault-tolerant strategies, i.e., the redundancy strategy and the degradation strategy, have attracted great research attention. The main difference between the two strategies is that the redundancy strategy generates logical arrays with given size in order to meet the requirement of particular applications while the degradation approach constructs logical arrays with size as large as possible.

For degradation strategy, many reconfiguration algorithms have been proposed under three different rerouting constraints, they have proved that most problems that arise under the three constrains are NP-complete [2]. The work in [3] simplified the general reconfiguration problem to *reconfiguration on selected rows* with limitation on routing distance and it is optimally solved by an algorithm named GCR (Greedy Column Rerouting). An efficient algorithm is introduced to reduce power dissipation of a logical array in [4], combined with GCR algorithm by reducing the number of long-interconnects. In addition, the work in [5], [6] extends the reconfiguration problem to three-dimensional processor arrays with efficient reconfiguration algorithms.

In practice, some particular applications require a processor array of a certain size. The redundancy strategy aims at constructing a logical PE array of a certain size to provide the same functionality with no slowdown. For redundancy strategy [7], a certain amount of redundancy PEs are integrated in physical array. If some PEs in circuit fails to work, these faulty PEs are replaced by the reserved redundancy PEs. The work in [8] proposes a self-repairing circuit based on mesh-connected processor array, and uses advanced integrated redundancy PE to replace the faulty PE. And the work in [9] showed the feasibility of the redundancy strategy. This paper also follows the redundancy strategy for the reconfiguration problem.

There are still different switching mechanisms in the redundancy reconfiguration, such as the work in [10] by selecting the multi-track switch to improve the reconfigurable rate of the array. Here multi-track means more than one communication path along each horizontal/vertical channel (The communication path includes both the data and control signals between two adjacent PEs). However, the single-track switch [4] is obviously save the area of hardware, so this paper also uses the single-trick switch like most previous studies.

In recent years, torus network is attracting more and more attentions [11], [12], [13]. It is utilized in designing on-chip networks (NoC), reconfigurable system, etc. Moreover, torus network based routing algorithms are widely investigated. However, to the best of our knowledge, we are not aware of any reported work on fault-tolerance of torus connected processor arrays. Motivated by this, we applied the contradiction graph approach to the reconfiguration of torus connected processor array. In this paper, we model the problem of replacing faulty PEs with spare ones as the maximum independent set problem on contradiction graph. We also design efficient algorithms for constructing contradiction graphs from physical arrays with faulty PEs, finding maximum independent set on contradiction graphs, and generating logic arrays based on the maximum independent set.

2 Preliminaries

This section presents definitions and important theories that will be useful for subsequent sections.

2.1 Fault-Tolerant Architecture

We introduce two kinds of arrays: physical array and logical array. A physical array H is the original array after manufacturing in which some of its PEs are defective. Let N_f be the number of faulty PEs in physical array H with size of $m \times n$. A logical array T is a subarray of H after reconfiguration such that T contains no faulty PE. The rows (columns) in the physical array and logical array are called the physical rows (columns) and logical rows (columns), respectively. Throughout this paper, (r, c) indicates a PE of physical array H where r is its row index and c is its column index in H, and the physical array is indexed from [0, 0] to $[m-1, n-1]$.

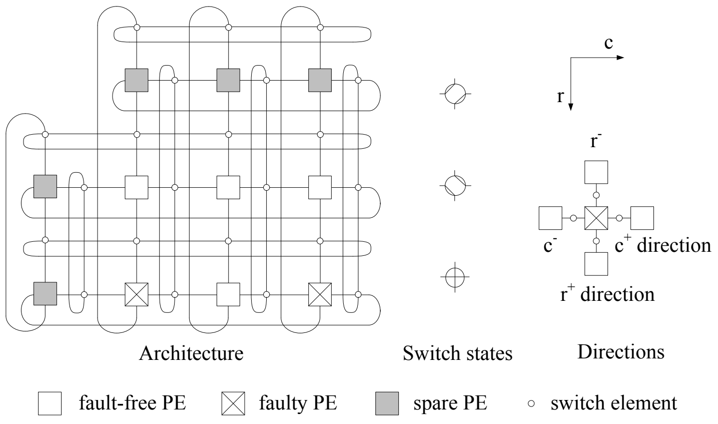


Fig. 1. The reconfigurable multiprocessor system architecture

The physical array used in our discussion consists of PEs, switching elements and interconnection wirings. Fig. 1 shows a 3×4 torus-connected processor array with one row and one column of spare PEs. Each square box in the host array represents a PE, while each circle represents a configuration switch. Adjacent PEs are connected by links with four-port switches. There are three states for each switch, and these states can be switched according to the needs of the array. The fault-tolerant reconfiguration is achieved by inserting several switches and connections in the network allowing the network to dynamically change the connections among PEs. Such as [14], all switches and interconnects in an array are assumed to be fault-free as they form very simple structure in relation to the PEs. By the way, all spare PEs are considered to be fault-free and faulty PEs can be converted into connecting elements in this paper.

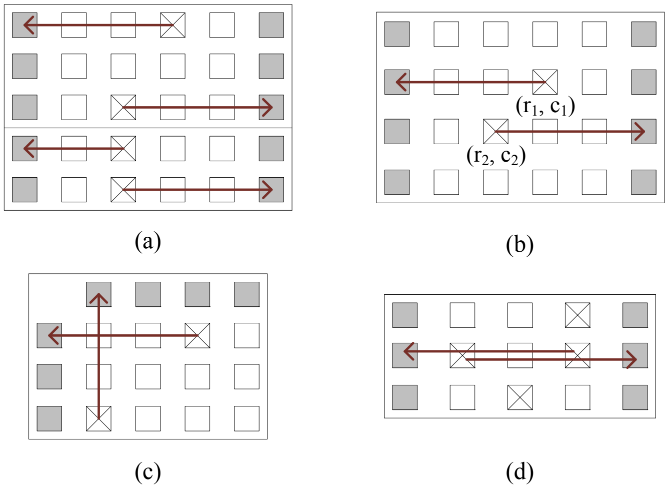


Fig. 2. Two kinds of replacement paths (a) not considered near-miss, (b) near-miss, (c) not in same row and column, (d) in same row

To better understand this paper, we introduce the following definitions.

1) *Replacement path.* If a PE, say (r, c), is detected to be faulty, then it might be replaced by a fault-free PE, say (r_1, c_1), which will in turn be replaced again by another PE (r_2, c_2), and so on. Eventually this replacement process terminates when a spare PE is used. This process defines the replacement path as the ordered sequence of physical PE, (r, c), (r_1, c_1), (r_2, c_2), $\cdots$. And the direction of the replacement path is defined to be the direction from faulty PE to spare PE in the replacement path. A faulty PE may have 4 possible replacement paths, in directions of upward, leftward, downward and rightward, respectively. As shown in Fig. 1, r (c) means the rows (columns) of the array. If PE (r_1, c_1) is replaced by a upward PE i, then the corresponding replacement path is denoted by $[r_1^-, c_1]$, and $[r_1^-, c_1]$ represents the number of rows to reduce while the number of columns unchanged, i.e., the replacement paths along with the r^- direction. For the

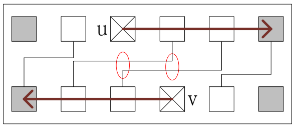


Fig. 3. The near-miss of replacement path with opposite directions

distribution of spare PEs shown in Fig. 1, each faulty PE, say (r, c), has four possible replacement paths, which are denoted as $[r^-, c]$, $[r^+, c]$, $[r, c^-]$ and $[r, c^+]$, respectively.

2) *Near-miss.* Two horizontal replacement paths, $[r_1, c_1^-]$ and $[r_2, c_2^+]$, are called near-miss if $|r_1 - r_2| = 1$ and $c_1 > c_2$ (see Fig. 2(b)). The near-miss caused by two vertical replacement paths of opposite direction can be similarly defined. Be noticed that the first row (column) and the last row (column) are adjacent rows (columns). All the other situations are not considered near-miss, as shown in Fig. 2(a).
3) *Intersection.* The intersection is defined to be the overlap of two replacement paths. As shown in Fig. 2(c) and (d).
4) *Contention.* The contention relation is caused by two replacement paths on the same row or on the same column, the two paths directed to the same spare PE thus cannot coexist.

2.2 Problem Description

Problem $\mathcal{R}$. *Given a $m \times n$ torus-connected processor array H with spare PEs, some PEs of H are defective due to fabrication or system usage, construct a logical torus array of given size by replacing faulty PEs using spare ones.*

In order to construct a feasible logic array, any faulty PE of the original physical array must be able to be replaced by a spare PE (directly or indirectly) through replacement paths without conflicts. In a feasible logical array, neither intersection nor near-miss among replacement paths is allowed, because they can not be implemented using multi-track switches. Fig. 3 shows an example of near-miss, the two replacement paths are rightward and leftward, respectively. In comparison to mesh array, the torus network has smaller diameter without increasing the node degree, due to the wraparound edges in the horizontal direction and vertical direction. Also, because of the same reason, the replacement path could be bent. However, a faulty PE cannot be replaced by a spare PE if the faulty one and the spare one are neither on the same row nor on the same column. We model the problem of whether all faulty PEs can be replaced by spare ones as maximum independent set problem on contradiction graphs which are constructed from the original physical array with faulty PEs. We next present the methods of generating contradiction graphs, then we present the algorithm for finding maximum independent set and the algorithm for calculating the logical array based on the independent set.

3 Generating Contradiction Graphs

We now present how to construct contradiction graphs from physical arrays with faulty PEs. In this paper, we consider three types of distribution of spare PEs. For a fauty PE i, it can be replaced by a non-faulty PE from left, right, top or down direction, respectively. We denoted the replacement path of PE i by a left-side (right-side, top-side or down-side) fault-free spare PE as v_i^l (v_i^r, v_i^t or v_i^b). In another word, v_i^t, v_i^b, v_i^l and v_i^r indicates that the directions of replacement paths are up, down, left and right, respectively. It is evident that v_i^t, v_i^b, v_i^l and v_i^r cannot coexist because each faulty PE i can only be replaced in one direction in a feasible logical array. Moreover, replacement paths with near-miss relation, intersection relation or contention relation cannot coexist. Let V_i be the set of all possible replacements of PE i. We construct the contradiction graph $G(V, E)$ where vertices set $V = \{V_i \mid i \in H \wedge i \text{ is faulty}\}$, and edge set E $=\{(u, v) \mid u \in V, v \in V, u$ and v can not coexists $\}$.

3.1 On Array with Two-Row Spare PEs

Fig. 4(a) shows the physical array with two-row spare PEs where gray boxes represent spare PEs. Since the two redundant rows of spare PEs are located at the top and bottom of the physical array, thus each faulty PE i has two possible replacement paths, i.e., v_i^t or v_i^b. The array in Fig. 4(a) contains 5 faulty PEs and each faulty PE has two possible replacement paths, thus there are 10 vertices in the contradiction graph.

The contradiction graph can be generated as follows. For every faulty PE i in H, the two vertices v_i^t and v_i^b are added into the vertices set V, where v_i^t and v_i^b denote the replacement of PE i by a fault-free PE from up-side and down-side, respectively. Edge set E is initialized to $\emptyset$, and is then constructed as follows.

1) There exists an edge between the two possible placement vertices of a faulty PE, i.e., there is an edge between vertices v_i^t and v_i^b for $1 \leq i \leq N_f$.
2) For any near-miss between two replacement vertices u and v ($u \in V$, $v \in V$), we add an edge between vertices u and v. Here we only need to consider the near-miss among two vertical replacement path, including the situation of two faulty PEs in the first column and the last column, respectively.
3) Let P_u and P_v denote two replacement paths, and u and v are two corresponding replacement vertices. If P_u overlaps with P_v, i.e., there is a intersection between P_u and P_v, we add an edge between vertices u and v. Here we only consider the intersection of two faulty PE on the same column.

3.2 On Array with One-Row-One-Column Spare PEs

Fig. 5(a) shows a physical array with one-row-one-column spare PEs, thus each faulty PE has four possible replacement paths. The array in Fig. 5(a) contains

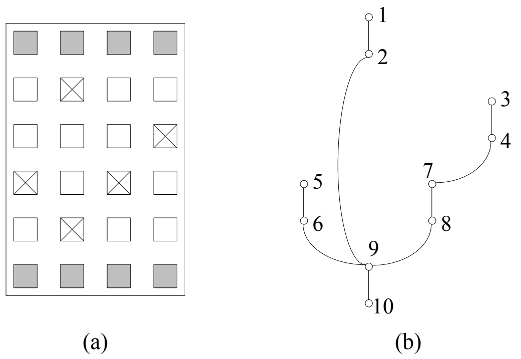


Fig. 4. The array with two-row spare PEs (a) physical array, (b) contradiction graph

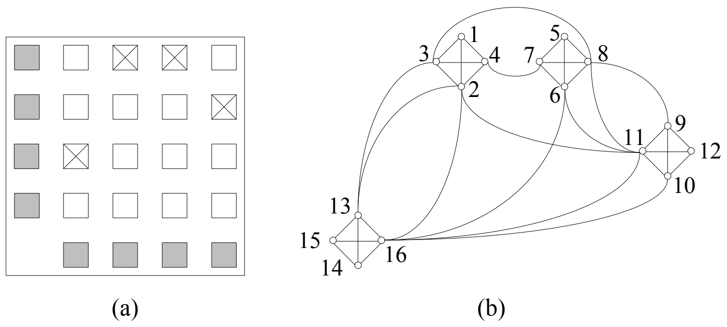


Fig. 5. The array with one-row-one-column spare PEs (a) physical array, (b) contradiction graph

four faulty PEs, thus the constructed contradiction graph comprises 16 replacement vertices, as shown in Fig. 5(b).

For each faulty PE i of H, four replacement vertices v_i^t, v_i^b, v_i^l and v_i^r are added into the set of vertices set V. The edge set E is initialized to $\emptyset$, and is then constructed as follows: There is an edge between any two vertices of v_i^t, v_i^t, v_i^t and v_i^b for $1 \leq i \leq N_f$. Let P_u and P_v denote two replacement paths, and u and v are two corresponding replacement vertices. If there is intersection, near-miss or contention relation between P_u and P_v, we add an edge between u and v.

3.3 On Array with Cross-Distributed Spare PEs

Fig. 6(a), the physical array with cross-distributed spare PEs. The physical array is distributed into four zones by the spare PEs. We denote the zones as I, II, III, IV. The contradiction graph contains 16 nodes as shown in Fig. 6(b).

We construct the contradiction graph in the following way. For each faulty PE i of H, the four vertices v_i^t, v_i^b, v_i^l and v_i^r are added to the vertices set V. Edge set E is initialized to $\emptyset$, and is then constructed as follows. Let P_1 and P_2

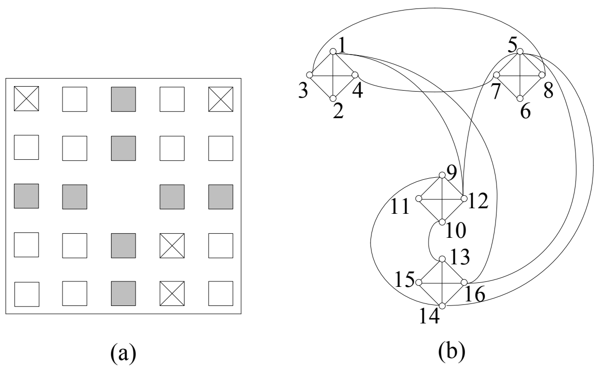


Fig. 6. The array with cross spare PEs (a) physical array, (b) contradiction graph

denote two replacement paths for two faulty PE (r_1, c_1) and (r_2, c_2), and u_1 and u_2 are the two corresponding replacement vertices in the graph.

1. If PE (r_1, c_1) and (r_2, c_2) are in a same zone (I, II, III or IV), we add edges using the same methods as in previous section.
2. Otherwise, if PE (r_1, c_1) and (r_2, c_2) are not in a same zone, we should pay attention to near-miss and intersection. The near-miss about two faulty PEs are not in the same zone can be similarly defined, such as when two faulty PEs, (r_1, c_1), (r_2, c_2), are not in the same zone, and if $|r_1 - r_2| = 1$ and $c_1 < c_2$, then two horizontal replacement paths, $[r_1, c_1^-]$ and $[r_2, c_2^+]$ are called near-miss. And the following combinations of zone should be consider about the intersection: two faulty PEs respectively in zones of (I, II) or (III, IV); two faulty PEs respectively in zones of (I, IV) or (II, III); two faulty PEs respectively in zones of (I, III); two faulty PEs respectively in zones of (II, IV).

4 Generating Logic Array Using Independent Set

4.1 Find the Maximum Independent Set

For a contradiction graph constructed from a physical array with N_f faulty PEs, if an independent set of N_f elements can be find, then all faulty PEs can be replaced by spare PEs (directly or consequentially), thus a feasible logical array can be formed. To find the maximum independent set of a contradiction graph, we customized a max-min ant system (MMAS) [15] algorithm in this subsection. MMAS have three key elements: *heuristic information*, *probabilistic decision* and *pheromone update*. MMAS starts from an empty solution and then iteratively chooses the best solution from all ants' solutions until the number of iterations is satisfied. In each iteration, after all ants generate their feasible solutions, we choose an ant with best solution. Then the ant deposits more pheromone on solution components which are used in constructing its solutions.

Solution components with larger amount of pheromone are more likely to be used by ants in constructing feasible solutions. The process repeats for a certain number.

In this paper, the solution components are the vertices of the contradiction graph, and the pheromone of a solution component corresponds to the probability that the solution component will be used by ants in constructing feasible solutions. We next present how to produce an independent set of a contradiction graph for an ant t. The sets are utilized: candidate set C_t and independent set S_t. C_t is initialized as all vertices of the contradiction graph and S_t is initialized as empty set. Algorithm MMAS randomly selects a vertex and removes it to S_t, then all its adjacent vertices of C_t will be deleted. Then MMAS repeatedly removes a vertex, say v, from C_t to S_t based on the *probabilistic decision* of each solution components, and deletes all neighbors of v from C_t. This process continues until C_t is empty, then the set S_t is the independent set for ant t.

– *Heuristic information.* Let $degree(v_i)$ be the degree of vertex v_i in the graph G. Then the heuristic information of vertex v_i is defined as $\eta(v_i) = 1/degree(v_i)$. The vertex with small heuristic information are more likely to be selected and removed from C_t to S_t.
– *Probabilistic decision.* Let $p^t(v_i)$ denotes the probability of vertex v_i being selected by ant t. The $p^t(v_i)$ is calculated as follows, based on pheromone trail $\tau(v_i)$ and heuristic information $\eta(v_i)$, where $\tau(v_i)$ will be defined later.

$$p^t(v_i) = \frac{[\tau(v_i)]^\alpha[\eta(v_i)]^\beta}{\sum_{v_j \in C_t}[\tau(v_j)]^\alpha[\eta(v_j)]^\beta} \tag{1}$$

where α and β are two parameters which determines the relative importance of the pheromone trail and the heuristic information.
– *Pheromone update.* At the end of each iteration, the pheromone trail is updated after all ants have generated their solutions. In order to simulate the pheromone evaporation, the pheromone on each vertex are multiplied by a residual concentration factor $(1-\rho)$, where ρ is the evaporation coefficient $(0 \leq \rho \leq 1)$. In addition, the ant with iterative best solution, say t_{ib}, will release pheromone on all vertices of its maximum independent set $S_{t_{ib}}$. The updated pheromone for each vertex $v_i \in S_{t_{ib}}$ is calculated as

$$\tau(v_i) = (1-\rho) \times \tau(v_i) + 0.1 \times degree(S_{t_{ib}}) \tag{2}$$

where $0.1 \times degree(S_{t_{ib}})$ indicates the pheromone deposited by t_{ib}. And vertex v_j outside $S_{t_{ib}}$ is calculated as

$$\tau(v_j) = (1-\rho) \times \tau(v_j) \tag{3}$$

And after a few iterations, such as 5 iterations, the global best ant t_{gb} released pheromone on the vertices of its maximum independent set $S_{t_{gb}}$, and the updated pheromone is defined as follows.

$$\tau(v_i) = (1-\rho) \times \tau(v_i) + 0.1 \times degree(S_{t_{gb}}) \tag{4}$$

where $0.1 \times degree(S_{t_{gb}})$ denotes the pheromone deposited by t_{gb}.

4.2 Generating the Logic Array

Algorithm 1. LogArr(H, N_f, *path_dire*)

```
Input: physical array $H$; the number of faulty PE $N_f$; the directions of
       replacement paths $path\_dire$;
Output: logical array $T$.
begin
$T := H$;
for each non-spare PE $(i, j) \in H$ do
    $i^p := i$; $j^p := j$;
    if $(i, j)$ on a replacement path, say $p_k$ then
        if $path\_dire(k)$ is upward then
            $i^l := (i^p - 1 + m)\%m$;
        else if $path\_dire(k)$ is downward then
            $i^l := (i^p + 1)\%m$;
        else if $path\_dire(k)$ is leftward then
            $j^l := (j^p - 1 + n)\%n$;
        else
            $j^l := (j^p + 1)\%n$;
    else
        the logical index of $(i, j)$ is its physical index;
return $T$;
end
```

Let *path_dire*(k) be the direction of replacement paths p_k, we initialize the *path_dire*(k) for each replacement path p_k according to the maximum independent set produced in previous stage. The proposed algorithm for generating logic arrays, denoted as `LogArr`, works in the following way. The `LogArr` starts by initializing the logic array as the physical array. Then, for each non-spare PE, say (i, j) ($0 \le i \le m - 1$, $0 \le j \le n - 1$), which is not located on a replacement path, `LogArr` sets the logical coordinate as the physical index of the PE (i, j). Otherwise, (i, j) located on a replacement path p_k, and we calculate the logical coordinate of the PE (i, j) as following. In order to distinguish, the row (column) index of a PE (i, j) in physical array H and logical array T is denoted as i^p (j^p) and $i^l(j^l)$, respectively.

1. If the direction of p_k is upward, $i^l := (i^p - 1 + m)\%m$, $j^l := j^p$;
2. If the direction of p_k is downward, $i^l := (i^p + 1)\%m$, $j^l := j^p$;
3. If the direction of p_k is leftward, $j^l := (j^p - 1 + n)\%n$, $i^l := i^p$;
4. If the direction of p_k is rightward, $j^l := (j^p + 1)\%n$, $i^l := i^p$.

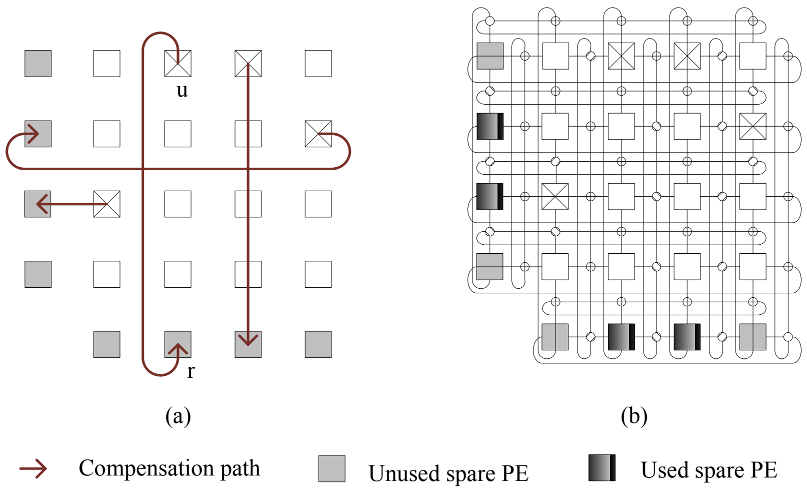


Fig. 7. The one-row-one-column distribution (a) the direction of replacement path, and (b) the connection of array

Due to all algorithms discussed in this paper are based on the torus-connected network, and the wraparound edges in the horizontal direction and vertical direction, we need to consider the modulo operation when calculating logical coordinate. Note that, for the two-row distribution of spare PEs, we only need to consider replacement paths of vertical direction.

To better understand how `LogArr` works, we employ an example that generating a logic array where the distribution of spare PE is one-row-one-column as shown in Fig. 7. In Fig. 7(a), the PE r is in the replacement path P_u of PE u, and u is replaced by r. The states of switches are shown in Fig. 7(b).

5 Experimental Results and Analysis

In this section, we evaluate the efficiency of the proposed reconfiguration approach for the fault-tolerance of torus array. The proposed algorithms were implemented in C language and ran on a Pentium IV 3.0GHz computer with 2GB RAM. The experiments are conducted on randomly generated data sets where the faults of a host array were generated by a uniform random generator the same as [3], [4]. In the experiments, the distribution of spare PEs are two-row distribution (case 1), one-row-one-column distribution (case 2) and the cross distribution (case 3), respectively. In order to evaluate reconfiguration performance, the *Reliability* is calculated as follows.

$$Reliability = \frac{\text{num of successful reconfigurations}}{\text{num of total reconfigurations}} \tag{5}$$

Table 1 shows the comparison of *Reliability* between mesh-connected array and torus-connected array with fault densities range from 0.5% to 5%.

The array size is set to be 16×16 and 32×32, respectively. Generally, for case 2 and case 3, the *Reliability* of torus-array is clearly higher than mesh-array. This is due to the fact that a spare PE can be utilized to replace more faulty ones in torus-array due to the wraparound edges, in comparison to mesh-array. In case of 16×16 host array with fault density 4%, the *Reliability* of torus-array is 1.000 while it is 0.567 for mesh-array in case 2, and the *Reliability* of torus-array is 0.967 while it is 0.833 for mesh-array in case 3. For case 1, not much improvement in *Reliability* is achieved in torus-array over mesh-array, because the more constraints is utilized in generating contradiction graphs for torus-array than mesh-array. For example, on 16×16 host array with fault density 5%, the *Reliability* of mesh-array is 0.367 while it is 0.333 for torus-array in case 1. In addition, few unexpected results are observed, such as the case on 32×32 host array with fault density 3%, the *Reliability* of mesh-array and torus-array are 0 and 0.033 in case 1. This is because, we find the maximum independent set by MMAS and MMAS with stochastic.

Table 1. The Impact of Fault Density to Reliability

Host Array		Reliability					
size	faulty density	case1		case2		case3	
		Mesh	*Torus*	*Mesh*	*Torus*	*Mesh*	*Torus*
16×16	0.5%	1.000	1.000	1.000	1.000	1.000	1.000
	1%	1.000	1.000	1.000	1.000	1.000	1.000
	2%	0.933	0.933	1.000	1.000	1.000	1.000
	3%	0.767	0.767	0.800	1.000	0.967	1.000
	4%	0.700	0.700	0.567	1.000	0.833	0.967
	5%	0.367	0.333	0.200	0.900	0.800	0.967
32×32	0.5%	1.000	1.000	1.000	1.000	1.000	1.000
	1%	0.967	0.967	0.733	1.000	0.967	1.000
	2%	0.333	0.300	0.033	0.833	0.500	0.833
	3%	0	0.033	0	0	0	0.033
	4%	0	0	0	0	0	0
	5%	0	0	0	0	0	0

For low fault densities, the *Reliability* of the three cases are all high. However, the *Reliability* of three cases decreases with the increasing fault densities. For example, for 32×32 host torus-array with fault densities increases from 2% to 3%, the *Reliability* of three cases decreases from 0.300 to 0.033, from 0.833 to 0, and from 0.833 to 0.033, respectively. This is because, the number of spare PEs is fixed, while the increased fault densities increases the number of faulty PEs, which increases the hardness of replacing all faulty PEs with spare ones.

Table 2 shows the comparison of *Reliability* between mesh-array and torus-array with array size ranging from 4×4 to 20×20. The fault densities of tested arrays are set to be 4% and 8%, respectively. It can be observed that the *Reliability* of torus-array is higher than that of mesh-array for case 1 and case 2. Also, few number of unexpected results are observed for case 1, due to the same reason as analyzed in Table 1. For small array size, the *Reliability* of the three cases are all very high, and the *Reliability* of the three cases are all decreases with the increasing array size. For example, on physical torus-arrays with fault density of 8%, *Reliability* of three cases decreases from 0.900 to 0.333, 1.000 to 0.767, 1.000 to 0.933, when the array size scales up from 8×8 to 12×12. It is because, with the increasing array size, but the number of faulty PEs increases quicker than the spare ones due to the distribution of spare PEs.

Table 2. The Impact of Array Size to Reliability

Host Array		Reliability					
faulty density	size	case1		case2		case3	
		Mesh	*Torus*	*Mesh*	*Torus*	*Mesh*	*Torus*
4%	4×4	1.000	1.000	1.000	1.000	1.000	1.000
	8×8	0.967	0.967	0.900	1.000	1.000	1.000
	12×12	0.733	0.733	0.833	1.000	0.967	1.000
	16×16	0.700	0.700	0.567	1.000	0.833	0.967
	20×20	0.300	0.300	0.100	0.833	0.200	0.900
8%	4×4	1.000	1.000	1.000	1.000	1.000	1.000
	8×8	0.900	0.900	0.867	1.000	1.000	1.000
	12×12	0.367	0.333	0.067	0.767	0.633	0.933
	16×16	0	0.033	0	0.167	0.033	0.267
	20×20	0	0	0	0	0	0

In comparison of the three cases, the distribution of case 2 and case 3 have higher *Reliability* than case 1, because each spare PE in case 1 can replace two faulty PEs at most, implying that the reconfiguration ability of case 1 is poorer. Combined Table 1 with Table 2, it can be concluded that, when the quantity of spare PEs is as much as twice of faulty ones, the two spare distribution patterns, the one-row-one-column pattern and the cross pattern, can provide stable fault-tolerance ability for torus-connected VLSI array.

6 Conclusions

In this paper, we have studied the fault-tolerance reconfiguration for torus-connected VLSI arrays. The reconfiguration problem is transformed into independent set on contradiction graph such that a physical array containing k faulty PEs can be reconfigured if an independent set of k elements can be found. We also presented efficient algorithms for constructing contradiction graphs from physical arrays with faulty PEs, for finding maximum independent set and for producing the target logical arrays based on the independent set. Simulation results indicate that, the PE array is of high reliability if the spare density is nearly twice as much as fault density of the physical array. In addition, the reconfiguration ability of the three proposed distribution patterns decreases with the increasing fault density and array size, thus new distribution pattern of spare PEs should be investigated for large arrays with dense faulty PEs.

References

1. Zhang, L., Han, Y., Xu, Q., Li, X., Li, H.: On Topology Reconfiguration for Defect-Tolerant NoC-Based Homogeneous Manycore Systems. IEEE Transactions on Very Large Scale Intergration (VLSI) Systems 17, 1173–1186 (2009)
2. Kuo, S.Y., Chen, I.Y.: Efficient Reconfiguration Algorithms for Degradable VLSI/WSI Arrays. IEEE Transactions on Computer-Aided Design 11, 1289–1300 (1992)
3. Low, C.P.: An Efficient Reconfiguration Algorithm for Degradable VLSI/WSI Arrays. IEEE Transactions on Computers 49, 553–559 (2000)
4. Wu, J., Srikanthan, T., Jiang, G., Wang, K.: Constructing Sub-Arrays with Short Interconnects from Degradable VLSI Arrays. IEEE Transactions on Parallel and Distributed Systems 25, 929–938 (2014)
5. Jiang, G., Wu, J., Sun, J.: Efficient Reconfiguration Algorithms for Communication-Aware Three-dimensional Processor Arrays. Parallel Computing 39, 490–504 (2013)
6. Jiang, G., Wu, J., Sun, J.: Non-Backtracking Reconfiguration Algorithm for Three-dimensional VLSI Arrays. In: 18th International Conference on Parallel and Distributed Systems, pp. 362–367. IEEE Computer Society Press, Singapore (2012)
7. Zhang, L.: Fault-Tolerant Meshes with Small Degree. IEEE Transactions on Computers 51, 553–560 (2002)
8. Takanami, I., Horita, T.: A Built-in Circuit for Self-Repairing Mesh-Connected Processor Arrays by Direct Spare Replacement. In: 18th Pacific Rim International Symposium on Dependable Computing, pp. 96–104. IEEE Press, Niigata (2012)
9. Banerjee, P., Peercy, M.: Design and Evaluation of Hardware Strategies for Reconfiguring Hypercubes and Meshes Under Faults. IEEE Transactions on Computers 43, 841–848 (1994)
10. Horita, T., Takanami, I.: Fault-Tolerant Processor Arrays Based on the 1.5-Track Switches with Flexible Spare Distributions. IEEE Transactions on Computers 49, 542–552 (2000)
11. Luo, W., Xiang, D.: An Efficient Adaptive Deadlock-Free Routing Algorithm for Torus Networks. IEEE Transactions on Parallel and Distributed Systems 23, 800–808 (2012)
12. Zhang, P., Powell, R., Deng, Y.: Interlacing Bypass Rings to Torus Networks for More Efficient Networks. IEEE Transactions on Parallel and Distributed Systems 22, 287–295 (2011)
13. Okazaki, R., Ono, H., Sadahiro, T., Yamashita, M.: Broadcastings and Digit Tilings on Three-Dimensional Torus Networks. Theoretical Computer Science 412, 307–319 (2011)
14. Wu, J., Srikanthan, T., Wang, X.: Integrated Row and Column Rerouting for Reconfiguration of VLSI Arrays with Four-Port Switches. IEEE Transactions on Computers 56, 1387–1400 (2007)
15. Li, Y., Xul, Z.: An Ant Colony Optimization Heuristic for Solving Maximum Independent Set Problems. In: 5th International Conference on Computational Intelligence and Multimedia Applications, pp. 206–211. IEEE Computer Society Press, Xi'an (2003)

An Ant Colony Optimization Algorithm for Virtual Network Embedding

Wenjie Cao, Hua Wang[*], and Lei Liu

School of Computer Science and Technology Shandong University
Jinan, Shandong Province, China
cwj@mail.sdu.edu.cn, {wanghua,l.liu}@sdu.edu.cn

Abstract. Virtual network embedding that embedding virtual network in substrate network is usually mentioned as resource allocation problem in network virtualization. Virtual network embedding can be employed to solve the problems like resource constraint, access control, request online and diversity of topology. This paper develops an ant colony optimization algorithm of virtual network embedding (ACO-VNE). The ants secrete and update pheromones in node mapping according to the cost of link mapping. Based on feedback information, the ants move to find good solution through learning from each other. Simulation results suggest that the algorithm can map the virtual network with low rejection rate and high revenue of substrate network.

Keywords: Network virtualization, virtual network embedding, path splitting, ant colony optimization.

1 Introduction

Network virtualization has been one of the most promising technologies for the future Internet [1]. Network virtualization allows coexistence of multiple virtual networks on the same physical network so that it can provide user with rich end-to-end custom services. Each virtual network (VN) in a network virtualization environment (NVE) is a collection of virtual nodes and virtual links. Essentially, a virtual network is a subset of the physical network (PN) resources. Network virtualization has been applied to the data center to solve problems such as extensibility, complexity, resource utilization, etc[2]. In addition, in order to achieve the sharing, separation and aggregation of computing resources and resource manageability in cloud computing environment, network virtualization becomes the key technology[3].

Virtual network embedding problem is core problem of network virtualization in resource allocation[4]. VNE focuses on the efficiency of embedding virtual network in physical network[5, 6]. Node resource usually includes CPU, memory, geographic location and link resource includes latency, bandwidth, jitter, etc. The aim of VNE is that embeds virtual network to the physical nodes and links appropriately and supplies

[*] Corresponding author.

enough resource for the virtual network. Under the guarantee of VN requests, VNE should reduce the rejection rate of virtual network request as much as possible and improve the revenue of physical network resources at the same time. VNE need to solve many problems, such as resource constraints, access control, online request and diversity of topology. VNE has been proven to be NP-hard problem[5].

As we know, virtual network embedding problem is very hard to find the optimal solution. However, ant colony optimization (ACO) as heuristic algorithm is powerful in solving NP-hard problem. ACO has been applied to multiple field like TSP, vehicle routing problem and multicast routing problem[7]. In this respect, we proposes ant colony virtual network embedding algorithm based on multi-constraints, denoted by ACO-VNE. ACO-VNE builds solution gradually by mapping virtual node to substrate node with low cost of mapping its adjacent links. The ant secretes and updates pheromone in the mapping of from virtual node to physical node according to the cost of embedding virtual network. From the learning between ants according to pheromone, the ant will gradually move closer to the optimum solution.

This paper consists of five sections. Section 1 briefs the background of the virtual network embedding. The related work of VNE is reported in section 2. The Mathematical Model is depicted in section 3. Section 4 briefs the ant colony virtual network mapping algorithm. The results of simulations and discussion of the performance are reported in section 5.

2 Related Work

The research of scholars in the past years has proposed many mapping scheme according to different direction. According to the difference of virtual network request (VNR) arriving, virtual network mapping problem can be divided into two cases: online and offline. Online VNE algorithm like [8-10] refers to that the virtual network request arrives at any time. PN will recover the network resources after the virtual network request leaves. While offline VNE algorithm like [11, 12] supposes knowing the arriving time, endurance and depart time of VNRs in advance. From the order of virtual node mapping and virtual link mapping, the VNE can be two stage VNE which virtual nodes are mapped before virtual links like[9, 10] or one stage VNE which virtual links are mapped at the same time as virtual nodes like [13, 14].

Early studies on VNE, greedy algorithm is one stage VNE mentioned in [9]. The algorithm always maps the virtual node with high resource demand to the substrate node with abundant resource. Then algorithm uses K-Shortest-Path (KSP) in link mapping. This algorithm can be accomplished simply, but it is difficult to obtain good solution because it can't make the utmost of the substrate resources of path with narrow bandwidth. So, another algorithm is proposed by [9] which sustained path splitting. Path splitting is that the virtual link can be mapped to multi-path in substrate network which is similar with multi-commodity flow[9]. The author proposed another way called path migration in link mapping for these VN requests with long hold-up time. Path migration allows a path to migrate from one substrate path to another path in order to save enough resource for the demand of current VN. But, it is difficult to determine which path to be migrated is best. Most methods of path migration is map VN again until the new VNR can be embedded into PN. Though path migration VNE can get a better solution than ordinary algorithm, but it needs higher time complexity.

For two stages VNE, the link mapping relies on the node mapping mostly. Based on the greedy idea of node mapping, [10] proposed VNE algorithm through topology-aware node ranking. Based on the Markov Random Walk model, the algorithm ranks the network node based on its resource and topological attributes. It computes the Node-Rank value for each virtual node and substrate node and then sorts virtual nodes in non-increasing order according to their Node-Rank values. Then the author proposed the link based breadth first search(BFS) based on backtracking. Once the failure of link mapping occurred, then it will go back to map the previous node. This algorithm leads to a good solution with a low time complexity. But when the network size is large, the solution of the algorithm is bad.

Above, we find that link mapping of VNE relies on node mapping largely. They are influenced with each other. One stage VNE can get better mapping result, but it using backtracking always results in bad time complexity. Two stages VNE, it is important to design a good node mapping algorithm. It is difficult to find optimal solution when the network scale is large because VNE is NP-hard problem. So, we propose an ant colony optimization for virtual network embedding.

3 The Mathematical Model of Virtual Network Embedding

Undirected graph $G_s(N_s, L_s, A_{N_s}, A_{L_s})$ denotes substrate network, where N_s denotes the node set of graph, each substrate node $n_s^i \in N_s$, i=$\{1,2,\cdots m\}$, m equals to $|N_s|$ means the number of nodes and L_s denotes the link set of graph, each substrate link $l_s^x \in L_s$, $x = \{1,2,p\}$, $A_{n_s^i}$ and $A_{l_s^i}$ demotes the property of n_s^i ($n_s^i \in N_s$) and l_s^x ($l_s^x \in L_s$) respectively. The property of n_s^i may be CPU, memory or location, while the property of l_s^x may be bandwidth. P_s denotes the path without loop.

Similar to the substrate network, the virtual network can be denoted $G_v = (N_v, L_v, C_{N_v}, C_{L_v})$, where N_v denotes the node set of graph, each virtual node $n_v^j \in N_v$ j=$\{1,2,\ldots$n$\}$, n equals to $|N_v|$ meaning the number of virtual nodes and L_v denotes the link set of graph, each virtual link $l_v^y \in L_v$, $C_{n_v^j}$ and $C_{l_v^y}$ demotes the resource demand of n_v^j ($n_v^j \in N_v$) and l_v^y ($l_v^y \in L_v$) respectively. Generally, we can use $V^{(j)}(G_v, t_a, t_l)$ to sign a VNR, where t_a denotes the time of VNE arrives and t_l denotes0 the time of VNE continues. The substrate network should provide enough resource like CPU, bandwidth to VNR when it arrives and take back resource when it departs. What's more, substrate network will deny the VNR without enough resource.

We define the mapping of VNE: $M: G_v(N_v, L_v) \to G_s(N_s{}', P_s{}')$, where $N_s{}' \in N_s$, $P_s{}' \in P_s$. Further on, we use $\gamma_j^i \in \{0,1\}$ to indicate if the virtual node n_v^j mapped into the substrate node n_s^i that means it is true if $\gamma_j^i = 1$, and vice versa. In the same way, we use $P_{l_v^y}$ denote the path which virtual link l_v^y mapped, and use θ_y^x denote the resource demand ratio that substrate link $l_s^x \in P_{l_v^y}$ occupied by $l_v^y \in L_v$. At first, we supposed each virtual node can only be mapped into one substrate node, in other words each virtual node can be supplied resource by one substrate node. Each substrate node can only supply one virtual node in the VNR. So, we get a constraint as follows:

$$\forall i \in \{1,2,\ldots,m\}, \quad \sum_1^n \gamma_j^i \le 1 \tag{1}$$

$$\forall j \in \{1,2,\ldots,n\}, \quad \sum_1^m \gamma_j^i = 1 \tag{2}$$

The substrate node which mapped by a virtual node must have enough resource. The constraint as follows:

$$\forall i \in \{1,2,\cdots,m\}, \forall j \in \{1,2,\cdots,n\}, \gamma_j^i (A_{n_s^i} - C_{n_v^j}) \ge 0 \tag{3}$$

The substrate link which mapped by a virtual link must have enough resource as well. The constraint as follows:

$$\forall l_v^y \in L_v, \forall l_s^x \in P_{l_v^y}, (A_{l_s^x} - \theta_y^x * C_{l_v^y}) \ge 0 \tag{4}$$

The goal of VNE is take full advantage of the physical network resource and reduce the reject rate of VNR as much as possible. That is, we should use resource as few as possible to fulfill the demand of VN. We only consider the link resource occupying because of the node resources occupying same from different mapping algorithm. So we define objective function as follows:

$$\min \left(\sum_{l_v^y \in L_v} \sum_{l_s^x \in P_{l_v^y}} \theta_y^x * C_{l_v^y}\right) \tag{5}$$

From the above, we get the mathematical model of VNE as follows:

The mathematic model of VNE

Minimize: $\sum_{l_v^y \in L_v} \sum_{l_s^x \in P_{l_v^y}} \theta_y^x * C_{l_v^y}$

Subject to:

$$\forall i \in \{1,2,\cdots,m\}, \sum_1^n \gamma_j^i \le 1$$
$$\forall j \in \{1,2,\ldots,n\}, \sum_1^m \gamma_j^i = 1$$
$$\gamma_j^i (A_{n_s^i} - C_{n_v^j}) \ge 0$$
$$\forall l_v^y \in L_v, \forall l_s^x \in P_{l_v^y}, (A_{l_s^x} - \theta_y^x * C_{l_v^y}) \ge 0$$
$$\gamma_j^i \in \{0,1\}, \theta_y^x \in [0,1]$$
$$\forall i \in \{1,2,\ldots,m\}, \forall j \in \{1,2,\ldots,n\}$$
$$\forall x \in \{1,2,\ldots,p\}, \forall y \in \{1,2,\ldots,q\}$$

4 ACO-Virtual Network Embedding Algorithm

In ACO-VNE, we suppose that one virtual node in same VN is not allowed to be mapped to the multiple substrate nodes in PN, while one virtual link can be mapped to multiple substrate links. Moreover, one substrate node only supplies one node in the same VN. In other words, different node from different VN can be embedded to the same substrate node. We suppose that the substrate network supports path splitting and uses multi-commodity flow algorithm to embed the virtual links to the substrate links[10].

The ACO-VNE algorithm based on ant colony optimization is a kind of one stage VNE that node mapping and link mapping are preceded simultaneously. For each VN, the ant will calculates transition probability to every substrate node for each virtual node through accumulating pheromone and heuristic factor. The probability will be zero if the substrate node has no enough resource or it has been mapped into anther virtual node in the same VN. Then, the ant selects the suitable node to be mapped using roulette algorithm. At last, the ant embeds the virtual links which adjoined the virtual node it mapped just now. The ant handles link mapping stage using multi-commodity flow(MCF) [9].

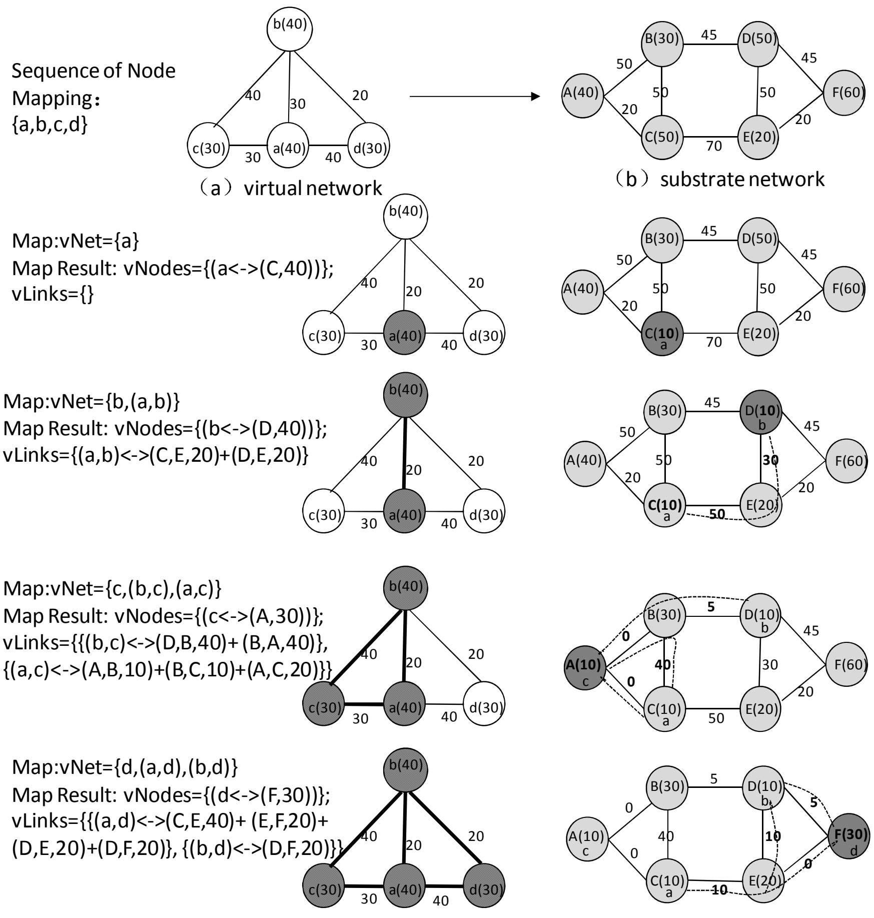
As shown in the fig 1, when a new VNR denoted VN_a arrives, PN_b marks the physical network. The numbers in parenthesis are the available CPU resources at the nodes and the numbers over the links represent available bandwidths. We only consider the CPU resource, geographic location of node and bandwidth resource of link. Considering the difference of resource demand of virtual nodes, the node with high demand can't be mapped successful when the substrate network is resource exhaustion. Obviously, the virtual node with high resource demand should be embedded prior. So, we sort virtual node according to their resource demand using the formula as formula (6), where L_{n_v} identify all adjacent links of virtual node n_v.

$$R_{n_v} = CPU(n_v) + \sum_{l_v \in \{L_{n_v}\}} BandWidth(l_v) \tag{6}$$

In the light of sequence of virtual node mapping $\{a,b,c,d\}$ in Fig1, we map fist virtual node a with no virtual link to be mapped now. Then, the ant maps the second virtual node b and its appending link (a,b) which connected with virtual node mapped before. The ant maps all the virtual nodes and virtual links in this way.

The ant selects candidate substrate nodes for a virtual node. The candidate node should be within the circular geographic area defined by the location of virtual node and its radius distance demand. Then, the ant will calculates transition probability to every substrate node for each virtual node through accumulating pheromone and heuristic factor through formula(7), where $\tau_{ij}(t)$ denotes the quantity of pheromone if n_v^i mapped into n_s^j in time t and η_{ij} is heuristic factor and α,β controls the influences of pheromone, heuristic factor.

$$p_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h \in N_s}[\tau_{ih}(t_s)]^{\alpha}[\eta_{ih}]^{\beta}}, \forall h \in N_s \\ 0,\ otherwise \end{cases} \tag{7}$$



Sequence of Node Mapping：{a,b,c,d}
(a) virtual network
(b) substrate network
Map:vNet={a}
Map Result: vNodes={(a<->(C,40))};
vLinks={}
Map:vNet={b,(a,b)}
Map Result: vNodes={(b<->(D,40))};
vLinks={(a,b)<->(C,E,20)+(D,E,20)}
Map:vNet={c,(b,c),(a,c)}
Map Result: vNodes={(c<->(A,30))};
vLinks={{(b,c)<->(D,B,40)+ (B,A,40)},
{(a,c)<->(A,B,10)+(B,C,10)+(A,C,20)}}
Map:vNet={d,(a,d),(b,d)}
Map Result: vNodes={(d<->(F,30))};
vLinks={{(a,d)<->(C,E,40)+ (E,F,20)+
(D,E,20)+(D,F,20)}, {(b,d)<->(D,F,20)}}

Fig. 1. The process of embedding VNR in ACO-VNE

In Fig1, if the ant maps virtual node *a* in substrate node *D* then the heuristic factor is the cost of mapping virtual link (*a*, *b*). The ant calculates heuristic factor using the formula(8), where L_{ij} marks all adjacent virtual link to be embedded if virtual node n_v^i have been embedded to node n_s^j. We define $Res(Map(l_v))$ as the resource occupied by the virtual link l_v using multi-commodity flow algorithm. Such as virtual link (*a*, *c*) in Fig1 is mapped to substrate link (*A*, *B*), (*B*, *C*) and (*A*, *C*). The bandwidth resource of (*A*, *B*), (*B*, *C*), (*A*, *C*) are occupied 10, 10, 20, by virtual link (*a*, *c*) with bandwidth demand of 30, respectively.

$$\eta_{ij} = \frac{1}{\sum_{l_v \in \{L_{ij}\}} Res(Map(l_v))} \tag{8}$$

We keep the best ant denoted A_{best} till all ants get result. The pheromone is stored in matrix $p[n][m]$, where *n* is defined the number of the virtual nodes and *m* is defined the number of substrate nodes. For each ant, we use the formula(9) update pheromone, where ρ denotes the heuristic factor and $\tau_{ij}(t)$ denotes the quantity of pheromone if n_v^i embedded into n_s^j in time *t*. We use formula (10) calculate τ_{ij}^l where *n* denote the number of ants.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{l=1}^{n} \Delta\tau_{ij}^l \tag{9}$$

$$\tau_{ij}^l = \frac{1}{\sum_{l_v \in L_v} Res(Map(l_v))} \tag{10}$$

When a virtual network request comes, we use ACO-VNE algorithm as follows:

Algorithm 1: ACO-VNE algorithm

```
1. Construct the sequence list of virtual node to be embedded in VN.
2. Initialize the pheromone matrix, transition probability matrix, the best ant
   $Ant_{best}$=+∞.
3. Input maximum iteration maxIte and the number of ants n
4. for i=0 to maxIte do
5.     for j=1 to n do
6.         Construct the mapping results $Map_j$ through algorithm 2
7.         if $Res(Map_{best}) > Res(Map_j)$ then
8.           $Ant_{best} \leftarrow Ant_j$
9.     end for
10.    Update pheromone matrix
11.    if converge then
12.      break;
13. end for
14. return $Ant_{best}$
```

For each ant, we use the algorithm to embed VN as follows:

Algorithm 2: ACO-VNE Mapping algorithm

```
1. Define the node, link mapping results of VNE NodeMap, LinkMap respectively
2. Select the first virtual node $n_v^0$ in sequence list
3. Select the substrate node $n_s$ with most resources
4. Update the NodeMap with <$n_v^0, n_s$>
5. for $n_v^j$ in $N_v \backslash n_v^0$ do
6.     Find the vLinks which adjacent to $n_v^j$ will be mapped
7.     Find the candidate substrate nodes $N_s$ which located in constraint location of $n_v^j$
8.     for $n_s^i$ in $N_s$ do
9.         Map vLinks through GLPK and get edge mapping edgeMap($n_s^i$) and the
   cost $Cost_{n_s^i}$
10.        Calculate the factor and update transition probability matrix
11.    end for
12.    Select $n_s$ through Roulette algorithm
13.    Update NodeMap, LinkMap
14. end for
15. return the solution of Embedding NodeMap and LinkMap
```

5 Simulation and Results Analysis

5.1 Simulation Environment

The simulation tool we use is Alevin that is a Java project with well-defined interfaces to implement new VNE algorithms and metrics. Alevin use Waxman generator to create random network topologies. ACO-VNE we proposed is appropriate for VNR offline. That is, we have known all VNRs before we run the algorithm. So, VNRs in our simulation environment is offline.

5.2 Evaluation Metrics

The goal of virtual network mapping is make full use of limited the underlying network resources and improve the revenue of infrastructure providers. So we define the formula (11) to evaluate revenue of VNR, where $C_{l_v^y}$ denotes the resource demand of link l_v^y and L_v denotes links of all VNR.

$$Rev = \sum_{l_v^y \in L_v} C_{l_v^y} \tag{11}$$

What's more, we use the formula(12) to evaluate utilization rate that virtual network mapped to substrate network, where L_v denote the set of links of VNR and $P_{l_v^y}$ denote the substrate path which embedded by virtual link l_v^y. $C_{l_v^y}$ denotes the resource demand of link l_v^y and θ_y^x denotes the ratio of resource demand applied by the substrate link l_s^x. R/C reflects the relation between the profit and the cost of VNR.

$$R/C = \frac{\sum_{l_v^y \in L_v} C_{l_v^y}}{\sum_{l_v^y \in L_v} \sum_{l_s^x \in P_{l_v^y}} \theta_y^x * C_{l_v^y}} \tag{12}$$

We use the evaluation metric rejection rate (RA) of VNR as follows:

$$RA = \frac{N_v}{N} \tag{13}$$

Where N_v denotes the number of VNRs which mapped into substrate network failed. N denotes the number of all the VNRs.

5.3 Evaluation Results

In Alevin, we implement the ACO-VNE. In order to test performance of the algorithm, we implement K-Shortest-Path algorithm(KSP-VNE) , sustaining path splitting algorithm(SPL-VNE)[9].

From Fig2, we find that the three algorithms get a low RA when the number of VNRs is few. However the RA of KSP-VNE increased quickly along with more VNRs. Because KSP-VNE do not allowed path splitting when there are many VNRs. The RA of ACO-VNE is not higher than SPL-VNE in any case. But it can’t search solution globally although SPL-VNE allows path splitting. Whatever, ACO-VNE based on swarm intelligence, it tend to search good results from the whole solution space.

From the Fig3, we find that the revenue of ACO-VNE is higher than KSP-VNE and SPL-VNE. Because of ACO-VNE embed virtual network to substrate network considering the embedding of future virtual network by feedback. It is obvious when there are more VNR from Fig3.

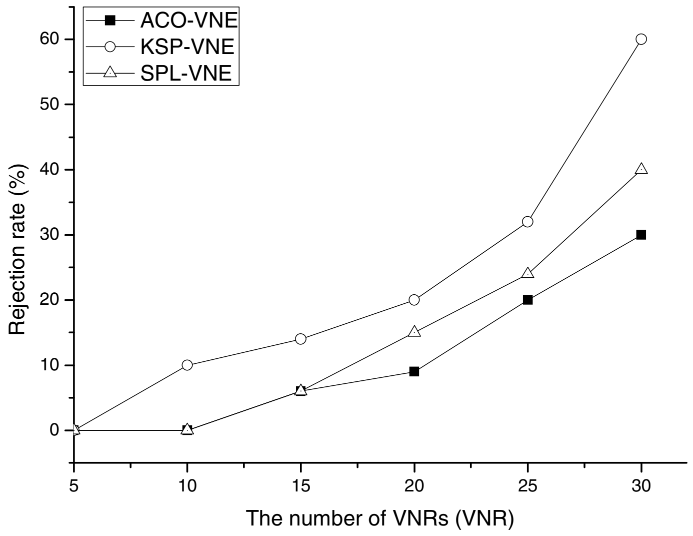


Fig. 2. Comparion of the rejection rate of the algorithms

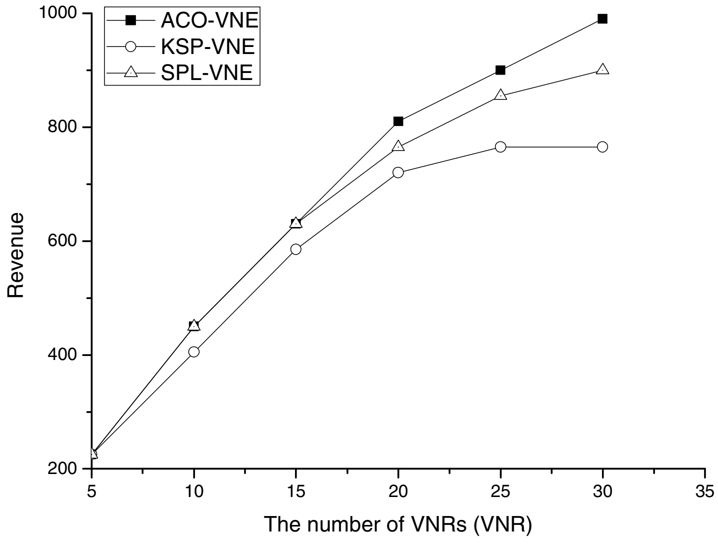


Fig. 3. Comparion of the revenue of the algorithms

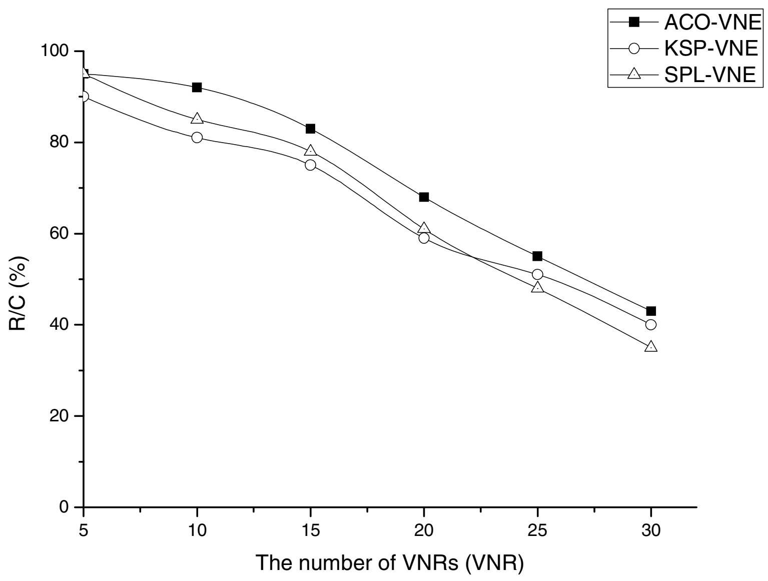


Fig. 4. Comparion of the R/C of the algorithms

From the Fig4, we find that the R/C of all three algorithms is high when there are few VNRs. When there are more VNRs, the R/C decreases because SPL must expend more resource such as long path, multi-path to satisfy the demand of new VNR. The R/C of ACO-VNE, SPL-VNE is higher than KSP-VNE because of path splitting. Furthermore, the R/C of ACO-VNE is higher than SPL-VNE. Because of ACO-VNE can find cheaper multi-path than SPL-VNE according to their learning from each other. In the fourth point, we find that the R/C of SPL-VNE is lower than KSP-VNE when there are more VNRs because KSP-VNE can't accept more VNR. While, SPL-VNE can accept more VNRs but it consumes more resource.

6 Conclusion

We propose ACO-VNE based on the ant colony optimization for virtual network embedding algorithm in this paper. The algorithm uses swam evolution idea and sustains splitting path as well. The ant embed virtual node according to the cost of embedding its adjacent links. The ant finds solution towards optimal solution stimulated by pheromone secreted by other ants. Simulation results indicate that ACO-VNE can get a better solution for VNR than traditional algorithm with lower rejection rate and higher revenue.

Acknowledgment. The study is supported by the Natural Science Foundation of Shandong Province (No. ZR2011FM021; ZR2013FM029; ZR2013FQ013), the Science and Technology Development Program of Jinan (No.201303010), the National Natural Science Foundation of China (NSFC No. 60773101), and the Fundamental Research Funds of Shandong University (Grant No. 2014JC037).

References

1. Hermann, A.B.A.F.: Virtualisierung im Future Internet Virtualisierungsmethoden und Anwendungen. Informatik-Spektrum 33, 186–194 (2010)
2. Bari, M.F., et al.: Data Center Network Virtualization: A Survey. IEEE Communications Surveys and Tutorials 15(2), 909–928 (2013)
3. Jain, R., Paul, S.: Network Virtualization and Software Defined Networking for Cloud Computing: A Survey. IEEE Communications Magazine 51(11), 24–31 (2013)
4. Chowdhury, N., Boutaba, R.: A survey of network virtualization. Computer Networks 54(5), 862–876 (2010)
5. Fischer, A., et al.: Virtual Network Embedding: A Survey. IEEE Communications Surveys and Tutorials 15(4), 1888–1906 (2013)
6. Qing, S.-D., et al.: Virtual network embedding algorithms in the network virtualization environment. Journal of Software 23(11), 3045–3058 (2012)
7. Wang, H., et al.: A tree-growth based ant colony algorithm for QoS multicast routing problem. Expert Systems with Applications 38(9), 11787–11795 (2011)
8. Chowdhury, M., Rahman, M.R., Boutaba, R.: ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. IEEE-ACM Transactions on Networking 20(1), 206–219 (2012)
9. Veitch, D.: Rethinking virtual network embedding: Substrate support for path splitting and migration. ACM SIGCOMM Computer Communication Review 38(2), 19–29 (2008)
10. Xiang, C., et al.: Virtual Network Embedding Through Topology-aware Node Ranking. Computer Communication Review 41(2), 39–47 (2011)
11. Houidi, I., et al.: A distributed Virtual Network mapping algorithm. In: Proceedings of the 2008 IEEE International Conference on Communications, vol. 1-13, pp. 5634–5640 (2008)
12. Shamsi, J., Brockmeyer, M.: QoSMap: QoS aware mapping of virtual networks for resiliency and efficiency. In: Proceedings of the 2007 IEEE GLOBECOMM Workshops, pp. 110–115 (2007)
13. Leivadeas, A., Papagianni, C., Papavassiliou, S.: Efficient Resource Mapping Framework over Networked Clouds via Iterated Local Search-Based Request Partitioning. IEEE Transactions on Parallel and Distributed Systems 24(6), 1077–1086 (2013)
14. Lischka, J., Karl, H.: A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection. In: Visa 2009, pp. 81–88 (2009)

Temperature-Aware Scheduling Based on Dynamic Time-Slice Scaling

Gangyong Jia[1,2], Youwei Yuan[1], Jian Wan[1,2], Congfeng Jiang[1,2], Xi Li[3], and Dong Dai[4]

[1] School of Computer Science, Hangzhou Dianzi University
Hangzhou, 310018, China
[2] Key Laboratory of Complex Systems Modeling and Simulation of Ministry of Education
(Hangzhou Dianzi University)
Hangzhou 310018, China
[3] School of Computer Science, University of Science and Technology of China
Hefei, 230027, China
[4] School of Computer Science, Texas Tech University
Lubbock TX 79409, USA
ganyong@mail.ustc.edu.cn

Abstract. As power density increases with high technology, the high temperature has threatened the system performance, reliability and even system safety. In this paper, we propose a temperature-aware task scheduling approach which combines low-overhead Time-Slice Scaling (TSS) with Alternative Scheduling schemes to reduce temperature. Through fine-grained thermal characterization based on task behavior, dynamically determine Time-slice Scaling factor (TSF) for each task on real-time. Besides, combining alternative scheduling scheme based on boosting thermal model. Experimental results demonstrate our proposed policy can reduce the chip average and peak temperature maximums by 3.1°C and 2°C with negligible performance loss.

1 Introduction

Thermal problem in modern microprocessor is becoming more and more serious with continuing decrease of device feature size, which results in more computation being processed in smaller area. If the cooling device can't efficiently scatter the generated heat, chip temperature will rise which causing more power dissipation and the thermal state will fall into a feedback loop and eventually leads to thermal runaway [18].

Although many architecture-level Dynamic Thermal Management (DTM) techniques have been proposed to alleviate processor's thermal emergency, these power-aware techniques control runtime thermal states at the cost of high performance loss and based architecture support. System-level DTM methods provide higher flexibility and scalability in solving thermal problems. Temperature-aware task scheduling is one of the most popular and well-studied system-level DTM methods. Such as R. Jayaseelan et al. proposed Temperature-aware task sequencing and voltage scaling policy [5], which combined alternative scheduling of hot and cool task with architecture supported Dynamic Voltage Frequency Scaling (DVFS) after the chip temperature exceeds threshold, and so on [7, 20]. All these policies control temperature using time-slice as the basic smallest unit. Therefore, if chip temperature exceeds threshold temperature within one time-slice, only two circumstances will happen: 1) use DVFS technology to decrease temperature but with considerable performance loss; 2) increase radiating to decrease chip temperature but with considerable more energy consumed.

In this paper, we propose a temperature-aware task scheduling approach based on more fine-grain unit of less than one time-slice. According to task thermal behavior characterization, dynamically determine Time-slice Scaling factor (TSF) for each task on real-time which hotter task will have less TSF, therefore, allocated less time-slice for original time-slice multiplied by TSF, and cooler task will be allocated more time-slice. Time-Slice Scaling can reduce temperature with neither performance nor energy overhead. Moreover, we design a boosting thermal model which predicts the temperature of the chip. Based on the boosting thermal model, determine the alternative scheduling scheme to reduce temperature further.

We implement the proposed scheduling approach in Linux 2.6.22 running on an Intel Core 2 Solo Processor. We choose 15 heterogeneous benchmarks from Mibench, SPEC 2000 and 2006, divide them into 14 experimental groups according to varied CPI characteristics combination. The experimental results show that our technique can reduce chip average and peak temperature maximums by 3.1◦C and 2◦C, and greatly shorten the time length of temperature peak with negligible performance loss.

The rest of this paper is organized as follows: in section 2, introduce some related work. Section 3 explains the method we used to characterize task's thermal characterization. Section 4 describes the temperature-aware scheduling based on Time-Slice Scaling and Alternative Scheduling schemes. We report the experiment setups and results in section 5 and conclude in section 6.

2 Background

Much previous work had explored various methods utilizing DVFS, DPM or other system-level policies to achieve dynamic power, energy and thermal management. The optimization objectives of these approaches can be classified into three categories: 1) runtime chip average and peak temperature minimization [4-6, 8, 9, 14, 16]; 2) performance maximization under certain thermal/energy constraint [13, 14, 17, 19]; 3) energy or power minimization under certain thermal constraint [1-3, 10, 12, 18].

Jayaseelan and Mitra [8] observed that different task execution sequences produce different chip peak temperatures and then they propose a task sequencing and voltage scaling algorithm to reduce the runtime chip peak temperature for a set of periodic tasks on embedded system. Chantem et al. [4] formulate the problem of task scheduling and assignment into mixed-integer linear programming and propose peak temperature minimization approach for hard real-time application on MPSoCs. In [14], Liang Xia et al. implement a thermal-aware scheduler which leverages the temporal and spatial heat slack among cores to migrate task from hot core to cooler core and assign a cool task to hot core or let it idle to balance the heat between these cores. Temperature-aware scheduling based on calculation of the post-thermal map has been done in work [6]. Choi et al. [5] propose another temperature-aware scheduling approach which changes the runtime workload timely to reduce the on-chip unit temperature. Considering the situation when CPU is full-loaded, a chip temperature reduction technique is proposed in [9].

In [15], Zhang et al. introduced the thermal-aware performance maximization problem. They provided approximation algorithm to minimize the completion time for a set of periodic tasks under thermal constraints. [13] deal with the problem of performance optimization under thermal constraints for real-time system. [1] utilize the makespan as measure to design speed scaling algorithm for energy and power reduction. Liu et al. [10] considers several optimization objectives for embedded system under real-time constraints, such as energy optimization, thermal optimization and thermal constrained energy optimization. Wang et al. [12] propose a analytical model to characterize the thermal and workload features and then present a thermal-aware scheduling approach to reduce power dissipation and temperature in data center. [3] solve the problem of energy minimization through temperature-aware dynamic voltage selection technique.

All of these proposed techniques aim to manage system energy, power and thermal states and achieve a tradeoff with performance loss. These temperature-aware approaches basically utilize a general guideline, the so-called hot-out and cool-in scheme, to achieve online thermal management. Although they are more or less effective to reduce the chip average or peak temperature, off-line and task-level scheduling [8] can't leverage the temporal heat slack inside a task or between tasks thoroughly.

The primary contributions of our work are as follows:

1) Control temperature in more fine-grain, within time-slice. Propose time-slice scaling factor for each task, which represents the thermal behavior of the task. According to the TSF, allocate time-slice for task;
2) Design a boosting thermal model which predicts the temperature of the chip based on sequence of the scheduling;
3) Combine Time-Slice Scaling and Alternative Scheduling in our temperature-aware scheduling, which reduces chip average and peak temperature with negligible performance loss.

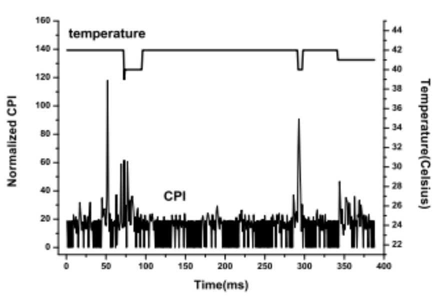
3 Thermal Characterization

Thermal characterization is the foundation for temperature-aware task scheduling. The execution flow of task is composed of alternative cpu-intensive and memory-intensive phases. In memory-intensive phases, the CPU obtains more opportunities for cooling. Therefore, the workload variation can cause chip temperature change. We perform experiments on Intel Core 2 Solo platform to observe task's workload characteristics and achieve thermal characterization through fine-grained workload characterization.

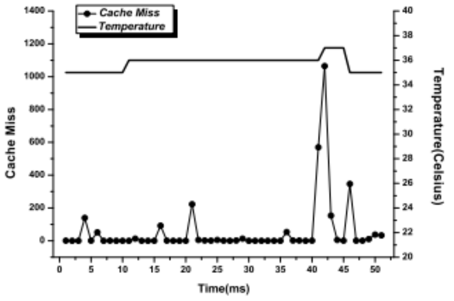
We run *susan* circularly and record the chip temperature and CPI every millisecond. Figure 1(a) shows the chip temperature and CPI variation over time under *susan*' steady temperature state. The variation of CPI (such as 50ms, 300ms and 340ms points) directly causes chip temperature change, although we can't observe more precise temperature change over time for the sensitivity limitation of on-chip temperature sensor supported by Intel Core 2 Solo process. High CPI normally implies low workload or utilization of CPU. Much prior work utilizes CPI and IPC or cache miss events to characterize and predict workload, which is inaccurate especially when predicting task's thermal characteristics in the near future. We run *tiffmedian* circularly and record the total number of cache miss and chip temperature every millisecond.

We intercept two execution phases (50ms) under stead temperature state and show them in figure 1(b) and 1(c). Table 1 shows the total number of cache miss in several 10ms-intervals and the temperature variation caused by cache misses. We observe that large number of cache miss cause temperature drop in the near future. In figure 1(b) and 1(c), if time 41ms is the scheduling point and we use CPI for workload prediction, the resulting predictions are the same in these two situations for that the CPIs are close (26.2 and 25.8 normalized). However, the actual situation is opposite that the temperature drops in Figure 1(b) after time 45ms point while rises after time 48ms point in Figure 1(c), which means that the workload characteristics of *tiffmedianin* the near future from prediction points are different in those two situations. CPI is not accurate enough for workload or thermal characterization and prediction because that it reflects the workload on phase-level while the lowering of chip temperature is mainly aroused by event-level cache miss. Therefore, the different cache miss distributions in a execution phase can indicate different workload features of a task in the near future, such as the large number of cache miss at time 41ms point in figure 1(b) indicates that it is extremely possible that there can be also large number of cache miss events in the near future. Therefore, for more accurate prediction, we need to investigate the cache miss distribution in the target time interval.

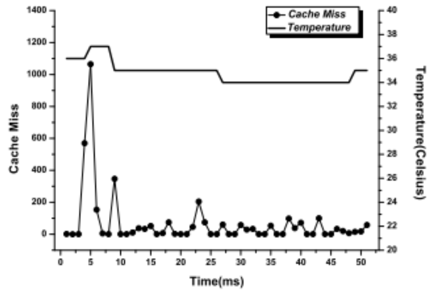
CPI is the average number of clock cycles needed for execution of one instruction. Ideally, the CPU commits per cycle while CPU usually stalls for various reasons such as cache miss, TLB miss and branch prediction miss. Therefore the actual CPI is usual larger than the ideal CPI. The ideal CPI can be calculated through formula 1.



(a) susan: CPI vs. Temp



(b) tiffmedian: CM vs. Temp



(c) tiffmedian: CM vs. Temp

Fig. 1. Thermal Characterization

$$CPI^{ideal} = CPI^{avg} - CPI^{avg}_{stall\_onchip} - CPI^{avg}_{stall\_offchip} \tag{1}$$

We collect the cache miss distribution information of the target time interval through online statistics supported by performance counters. We target the length of a scheduler quantum to calculate CPI and characterize cache miss distribution feature. The scheduler quantum (2n ms) is in the order of tens of milliseconds such as default 10ms in Linux. We use an array d=[d_1, d_2, ..., d_{2n}] to record the cache miss number of every millisecond in the latest scheduler quantum. The cache miss distribution (represented by φ) is expressed through formula 2. Now we combine CPI and cache miss distribution to characterize the thermal behavior for tasks. The thermal behavior (represented by Ψ) is calculated based on the formula 3.

$$\varphi = \begin{cases} 1 - \frac{\sum_{i=1}^{n} d_i}{\sum_{i=n+1}^{2n} d_i}, & \frac{\sum_{i=1}^{n} d_i}{\sum_{i=n+1}^{2n} d_i} \le 1 \\ \frac{\sum_{i=n+1}^{2n} d_i}{\sum_{i=1}^{n} d_i} - 1, & \frac{\sum_{i=1}^{n} d_i}{\sum_{i=n+1}^{2n} d_i} > 1 \end{cases} \tag{2}$$

$$\Psi = \min\left(\frac{CPI^{ideal}}{CPI^{avg}} * (1 - k\varphi), 1\right) \tag{3}$$

φ is a value between -1 to 1, which adjusts the value of CPI^{ideal}/CPI^{avg} dynamically. The larger the value of φ is, the cooler the target task is in the near future. The value of κ can be set between 0 and 1, the method for value setting is presented in [4].

Table 1. Tiffmedian: Temperature change after cache misses in 10ms-interval

	Time intervals (ms)				
	0-10	40-50	65-75	79-89	112-122
CM_{total}	110	89	43	54	1225
T_{var}	+1	+1	+1	+1	-2

	Time intervals (ms)				
	124-134	160-161	221-231	239-249	265-275
CM_{total}	637	596	45	6	18
T_{var}	-1	-1	+1	+1	+1

	Time intervals (ms)				
	281-291	312-322	373-383	407-417	421-431
CM_{total}	1034	328	5	32	527
T_{var}	-2	-1	+1	+1	-2

4 Temperature-Aware Scheduling

4.1 The Boosting Thermal Model

We formalize the problem of temperature-aware scheduling into a boosting thermal model as algorithm 1 shows. The boosting thermal model adopts the online learning theory. Online prediction is a continuous process, and at each trial point the online algorithm makes decision after receiving an N-component instance. Each component is a suggested prediction from an online learner. The learner's prediction is based on the input information and internal state of itself before the trial. The online algorithm evaluates each suggested prediction based on a loss function and chooses the learner's prediction with minimized loss.

In our boosting thermal model, we keep N learners and each one actually represents a task pool with the same scaled thermal behavior. At each scheduling point, the online learning algorithm evaluates each learner's suggestion (selecting one of the tasks represented by the learner who gives this suggestion) based on a loss function and choose the suggestion with minimized loss. We scale task's thermal behavior through formula 4 and associate a value ρ_j for learner j and let $\rho_1 = 1/(2N)$, $\rho_2 = 3/(2N)$, ..., $\rho_N = (2N-1)/(2N)$. Therefore, we associate learners to task pools with different scaled thermal behavior. We also associate and maintain an internal state weight vector for these learners, $v^t =< v^t_1 , v^t_2, ..., v^t_N>$,weight v^i_j represents the internal state of learner j at time t.

$$\Psi_s = \frac{\lceil 2N\Psi \rceil}{2N} \tag{4}$$

```
Algorithm 1: Temperature-aware scheduling algorithm
Parameters: $T_{amb}=T'_a$, $T_w=T'_w$, $c=c'$, $T_{tr}=T'_{tr}$, $N=N'$,
$\rho_i=(i+1)/2N'$
Initialize: weight vector $v^1(v_i^1 \in [0, 1])$
At scheduling point (time t):
1: Calculate $\Psi$, scale to $\Psi_s$ and record
2: Loss evaluation:
    $l_i^t = |\rho_i - \frac{T_t - T'_a}{T'_w - T'_a}|$
3: Update the weight vector:
    $v_i^{t+1} = v_i^t * (1 - (1 - \beta) * l_i^t)$
4: Choose suggestion:
    $w_k^t = \max\left(v_j^t / \sum_{i=1}^{N'} v_i^t\right)$
5: Task selection:
    $P^t(prio, \Psi_s) = P(prio_h, \frac{k+1}{2N'})$
6: if not in time-slice scaling:
    if $T_c > T'_{tr}$
        $S_t{*}=c$
        do time-slice scaling
    else $S_t = S_{default}$
7: t=t+1
```

The loss evaluation process is based on the real-time chip temperature at scheduling point and a heuristic rule (step 2 in algorithm 1). For leverage temporal thermal slack, temperature-aware scheduler chooses an appropriate cool task for execution when the chip temperature is high and vice verse. Non-temperature-aware scheduler may choose a hot task for execution when the chip temperature is high and a cool task when chip temperature is low. We identify these two situations as heat slack loss. Our online learning algorithm evaluates the loss factor l^t_i ($0 \le l^t_k \le 1$) at time t for learner i according to the formula 5.

$$l^t_i = |\rho_i - \frac{T_t - T_{amb}}{T_w - T_{amb}}| \tag{5}$$

Where T_t is the real-time chip temperature at time t, T_{amb} is the ambiance temperature and T_w is warning chip temperature or the temperature when running the hottest task. Once the loss factor is evaluated for each learner, the algorithm updates the weights of all learners through the formula 6 (step 3 in algorithm 1). The temperature at a time point is a function of all previous temperature states. Our online learning algorithm takes all previous temperature states into consideration to update the weights of learners. The value of β can be set between 0 and 1.

$$v^{t+1}_i = v^t_i * (1 - (1 - \beta) * l^t_i) \tag{6}$$

For choosing the suggestion with minimized loss, we need to scale learners' internal states. The algorithm maintains a vector $w^t = \langle w^t_1, w^t_2, ..., w^t_N \rangle$ to record the scaled value. The scaling process as formula 7 shows.

$$w^t_j = \frac{v^t_j}{\sum_{i=1}^{N} v^t_i} \tag{7}$$

At each scheduling point, the algorithm chooses the suggestion with highest probability factor among the all learners (step 4 in algorithm 1). Once the suggestion was taken, we obtain a candidate task pool to choose from for the next execution. In the actual situation, our scheduling is priority-based (such as the priority scheme in Linux), so the algorithm chooses the task (P^t) with highest priority ($prio_h$) in the task pool for execution in the next scheduler quantum (step 5 in algorithm 1). If the associated task pool is empty, the algorithm will choose a task with closest scaled thermal behavior.

4.2 Time-Slice Scaling (TSS)

Through experiments, we observe that the alternative execution of a hot task and an arbitrary cooler task for several short time intervals produces lower chip peak temperature than execution alone of the hot task for an extended time interval. We design a Time-Slice Scaling (TSS) scheme to adjust the length of time slice dynamically. When the chip temperature reaches the predefined temperature warning value, TSS scheme will be triggered (step 6 in algorithm 1). The Time-slice Scaling Factor (represented by c in the algorithm 1) is according to the thermal behavior of the task which is the formula 8. Because TSS scheme shortens the time slice of the hot task, therefore, we lengthen the time slice of the cool task for reducing both temperature and overhead of more switching frequency.

$$c = \frac{\sum_{i=n+1}^{2n} di}{\sum_{i=1}^{n} di} \tag{8}$$

5 Experimental Results

We implemented our temperature-aware scheduling in Linux 2.6.22 on an Intel Core 2 Solo processor. The algorithm frame is showed in figure 2. We modify the process control block *task_struct* to add a data field to record the scaled thermal behavior. We substitute the original multi-priority task queues with multi-priority 10-bucket hash tables and each bucket lists a task set with the same scaled thermal behavior. The internal state vector for online learners is implemented as a global array *inter_state* and we also create another global array *cm_distr* to record the number of cache miss in every millisecond of a scheduler quantum (default value is 10ms in Linux).

We use the on-die digital thermal sensor and interrupt mechanism supported by the Intel Core 2 Solo processor to report CPU temperature. In order to reduce overhead, we migrate the temperature sampling code of CoreTemp into kernel context (*scheduler_tick()*). In order to dynamically characterize thermal characteristics, we use the performance monitor units (PMU) supported by Intel Core 2 Solo Processor to record the number of cache misses, executed instructions and clock cycles.

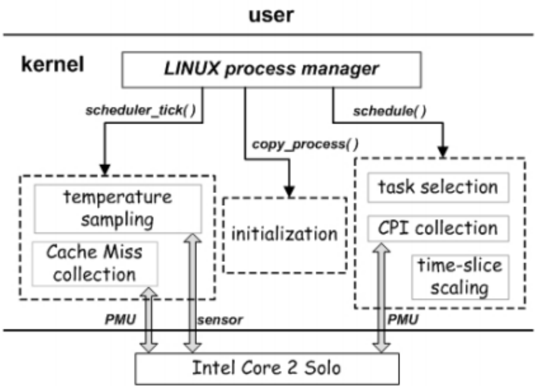


Fig. 2. Temperature-Aware Scheduling frame

We choose 15 benchmarks from Mibench and SPEC 2000 and show the 14 experimental groups in table 2. The third row in table 2 shows the CPI levels of these benchmarks where l<l+<m<*h*<*h*+. We run each benchmark in a group under the same initial chip temperature and compare the results under three cases (without TAS, TAS without TSS and TAS with TSS) in table 3. T_b, T_a, T_p, n, t denote the initial, average, peak temperature, number of occurrence of temperature peak and the length of execution time.

Table 3(a, b, c) displays our experimental results. We classify the 14 groups into three categories, g1~g4, g5~g10 and g11~g14. In category g1~g4, the chip average temperature is reduced by average 1.3 (case 2) and 1.7 (case 3) and the chip peak temperature is reduced by maximum 1 degree (g1, g2 and g4) or the time length of temperature peak is largely shortened (g3). In category g11~g14 which is hybrid, the chip average temperature is reduced by average 2.1 (case 2) and 2.8 (case 3) and the chip peak temperature is reduced by maximum 2 degree (g11). All benchmarks in category g1~g4 are from Mibench, which have shorter execution time and lower run-time temperature than benchmarks from SPEC 2000. And each group in category g1~g4 only includes 3 benchmarks. We observe task organization on priority hash tables in the runtime, it is usual that there is only one or two tasks in a priority hash table, which means that the temperature-aware task selection result has almost no difference with that of Linux default scheduler. The gains in chip average and peak temperature reduction is relatively smaller than case of category g11~g14. From the column of *delay*(%), we can see that the performance loss is relatively higher in category g5~g10 and g11~g14 than g1~g4. Longer execution time (more tasks or longer time of a single task) and higher runtime temperature can give more opportunities for temperature-aware scheduling and Time-Slice Scaling and Alternative Scheduling but introduce more extra activities then the higher delay. The delay in g14 reaches 11.3% while only 2.1 degree temperature reduction. High workload does not always mean positive effect on temperature reduction. The heterogeneity of workload is another important factor to affect our temperature-aware scheduling. The g6 is a combination of *l, m* and *h*+ and g7 is a combination of *m, h* and *l*, the average CPI of *mcf*(*h*+) is 19.07 which provides good heterogeneity for g6. The g6 reduces 1.5 degree more than g7.

Table 2. benchmark groups

Group	Mibench, SPEC2000 and SPEC2006	Description
g1	bitcount, tiff2rgba, dijkstra	h+, m, l
g2	rsynth, blowfish, sha	m, h, l+
g3	bitcount, dijsra, blowfish, sha	h+, l, h, l+
g4	bitcount, tiff2rgba, dijkstra, rsynth, blowfish, sha	h+, m, l, m, h, l+
g5	gzip, wupwise, art	m, l+, h
g6	gcc, mesa, mcf	l, m, h+
g7	eon, bzip2, apsi	m, h, l
g8	gzip, wupwise, gcc, mesa, art, mcf	m, l+, l, m, h, h+
g9	gzip, wupwise, art, eon, bzip2, apsi	m, l+, h, m, h, l
g10	gzip, wupwise, gcc, mesa, art, mcf, eon, bzip2,apsi	m, l+, l, m, h, h+, m, h, l
g11	bitcount, tiff2rgba, dijkstra, gzip, wupwise, mcf	h+, m, l, m, l+, h+
g12	bitcount, tiff2rgba, dijkstra, rsynth, blowfish, sha, gcc, mesa, art	h+, m, l, m, h, l+, l, m, h
g13	bitcount, tiff2rgba, dijkstra, rsynth, blowfish, sha, gzip, wupwise, gcc, mesa, eon, apsi	h+, m, l, m, h, l+, m, l+, l, m, m, l
g14	bitcount, tiff2rgba, dijkstra, rsynth, blowfish, sha, gzip, wupwise, gcc, mesa, art, mcf, eon, bzip2, apsi	h+, m, l, m, h, l+, h+, m, l, m, l+, h+

The delay is caused by several extra activities serving to our temperature-aware scheduling frame, such as online statistics, temperature sampling. We analyze the effect of these factors in the later section. The TSS mainly brings effect of two aspects, reducing the peak temperature directly or shortening the time length of temperature peak (g3's temperature peak length is shortened from 224 to 8). We turn off TAS and turn on TSS alone, and show the effect of TSS on peak temperature reduction in figure 3. The sub-figure above is in single task environment and the below one is in multi-task environment.

Table 3. (a) chip average/peak temperature and delay

Group	T_b	Case 1 (baseline)			
		T_a	T_p	n	t(s)
g1	37	43.5	45	243	20.94
g2	38	45	47	44	72.79
g3	37	44.9	46	224	26.93
g4	38	45.7	47	1085	93.63
g5	38	47.7	50	583	180.06
g6	36	47.1	49	28	185.88
g7	39	47.2	49	2	33.36
g8	40	49.9	51	5191	362.68
g9	39	49.4	51	1499	228.52
g10	38	49.6	51	5574	428.89
g11	37	48.1	50	3075	202.69
g12	37	47.1	49	457	272.38
g13	37	48.1	50	4832	389.17
g14	38	49.1	51	1	55.69

Table 3. (b) chip average/peak temperature and delay

Group	T_b	Case 2 (TAS without TSS)				
		T_a	T_p	n	$\triangle T_a$	delay(%)
g1	37	42.4	45	1	1.1	2.5
g2	38	44.1	46	29	0.9	3.2
g3	37	43.5	46	160	1.4	2.1
g4	38	43.8	47	10	1.9	4.4
g5	38	45.5	48	2144	2.2	6.7
g6	36	44.6	48	72	2.5	4.9
g7	39	46.1	49	1	1.1	3.9
g8	40	47.3	50	16529	2.6	6.8
g9	39	46.8	49	4515	2.6	7.1
g10	38	47.5	50	12	2.1	8.8
g11	37	45.8	49	684	2.3	5.9
g12	37	45.0	48	545	2.1	7.1
g13	37	45.7	49	4988	2.4	7.6
g14	38	47.5	50	17805	1.6	10.1

Table 3. (c) chip average/peak temperature and delay

Group	T_b	Case 3 (TAS with TSS)				
		T_a	T_p	n	$\triangle$	delay(%)
g1	37	42.1	44	391	1.4	2.7
g2	38	42.3	46	3	1.3	3.6
g3	37	43.3	46	8	1.6	2.6
g4	38	43.2	46	708	2.5	4.9
g5	38	45.2	48	1867	2.5	6.4
g6	36	44.0	47	4350	3.1	5.4
g7	39	45.6	48	395	1.6	4.3
g8	40	46.8	50	138	3.1	7.7
g9	39	46.4	49	1662	3.0	7.3
g10	38	47.1	49	8211	2.5	9.4
g11	37	45.1	48	2596	3.0	6.5
g12	37	44.2	48	98	2.9	7.8
g13	37	45.1	49	1811	3.0	8.2
g14	38	47.0	50	6562	2.1	11.3

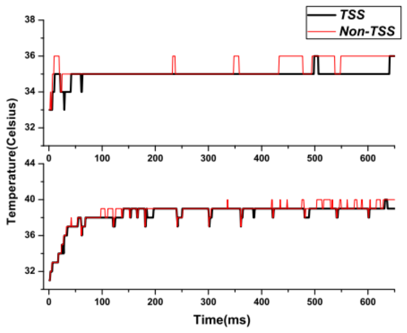
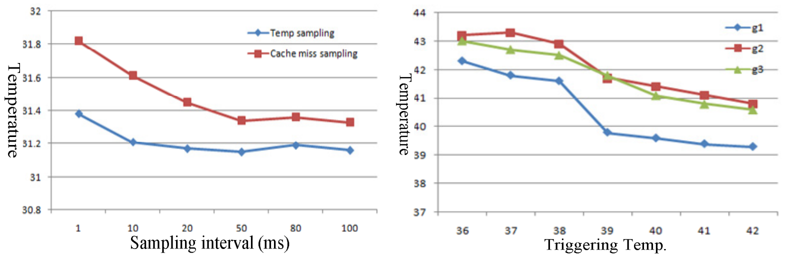


Fig. 3. mcf (T_{tr}=35◦C) and mcf+bzip (T_{tr}=39◦C)



(a) Overhead (b) Triggering Temperature

Fig. 4. overhead and triggering temperature

Overhead Analysis. The extra activities serving to our temperature-aware frame introduce overhead. Such as temperature sampling, CPI and cache miss collection, thermal behavior online characterization. In our implementation, we sample the chip temperature per 10 milliseconds and the number of cache miss per millisecond in clock interrupt handler *scheduler_tick()*. When CPU is idle, we sample temperature and cache miss number per 1, 10, 20, 50, 80 and 100 milliseconds and show the average temperature in figure 4(a) (sub-figure above). The overhead brought by temperature sampling and cache miss number collection is negligible relative to the reduction of chip temperature. Figure 4(a) (sub-figure below) also displays the average temperature under different scale levels. The overhead introduced by the time slice scaling is also negligible for bringing no more switching. We collect CPI and characterize thermal behavior in the order of tens of millisecond and the related code is short. Compared with cache miss number sampling and temperature sampling, the overhead of them is also negligible.

Parameters Analysis. Time-slice scaling can reduce peak temperature or shorten the time length of temperature peak. But the parameter of triggering temperature T_{tr} affects results. We analyze the effect using different T_{tr}. We run g1, g2 and g3 under different triggering temperature and show the average temperature in figure 4(b). When the triggering temperature is low, time-slice scaling scheme is triggered frequently, which reduces more temperature. As figure 4(b) shows, different groups have different optimal triggering temperatures. In our experiment, we determine the triggering temperature according to the peak temperature of a benchmark.

6 Conclusion

In this paper we propose a temperature-aware task scheduling approach based on more fine-grain unit of less than one time-slice. According to task thermal behavior characterization, dynamically determine Time-slice Scaling factor (TSF) for each task on real-time which hotter task will have less TSF, therefore, allocated less time-slice for original time-slice multiplied by TSF, and cooler task will be allocated more time-slice. Time-Slice Scaling can reduce temperature with neither performance nor energy overhead. Moreover, we design a boosting thermal model which predicts the temperature of the chip. Based on the boosting thermal model, determine the alternative scheduling scheme to reduce temperature further. Our proposed techniques achieve chip average and peak temperature maximums by 3.1◦C and 2◦C with negligible performance loss.

Acknowledgments. This work is supported by the National Science Foundation of China under grants (No. 61272131, No. 61202053, No. 61003077, No. 61202094), Zhejiang provincial Natural Science Foundation (No. LQ14F020011), Jiangsu provincial Natural Science Foundation (No. SBK201240198), Jiangsu production-teaching-research joint innovation project (No.BY2009128). We also gratefully acknowledge the support of our industrial sponsor, SAMSUNG (CHINA) R&D CENTER.

References

[1] Albers, L.A.S.: Power-aware scheduling for makespan and flow. In: Proc. of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 190–196 (2006)
[2] Bansal, N., Kimbrel, T., Pruhs, K.: Speed scaling to manage energy and temperature. J. ACM (2007)
[3] Bao, M., Andrei, A., Eles, P., Peng, Z.: Temperature-aware voltage selection for energy optimization. In: Proc. of DATE (2008)
[4] Chantem, T., Dick, R.P., Hu, X.S.: Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. In: DATE (2008)
[5] Choi, J., Cher, C.Y., Franke, H., Hamann, H., Weger, A., Bose, P.: Thermal-aware task scheduling at the system software level. In: Proc. ISLPED (2007)
[6] Cui, J., Maskell, D.L.: Dynamic thermal-aware scheduling on chip multiprocessor for soft real-time system. In: Proc. GLSVLSI (2009)
[7] Freund, Schapire: A decision-theoretic generalization of on-line learning and an application to boosting. JCSS (1997)
[8] Jayaseelan, R., Mitra, T.: Temperature-aware task sequencing and voltage scaling. In: Proc. of ICCAD (2008)
[9] Li, D., Chang, H.-C., Pyla, H.K., Cameron, K.W.: System-level, thermal-aware, fully-loaded process scheduling. In: Proc. IPDPS (2008)

[10] Liu, Y., Yang, H., Dick, R.P., Wang, H., Shang, L.: Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In: Proc. of ISLPED (2007)
[11] Severns, R.: Safe operating area and thermal design for mospower transistors. In: Siliconix Application Note (1983)
[12] Wang, L., Younge, A.J., Furlani, T.R., von, L.G., Dayal, J., He, X.: Towards thermal aware workload scheduling in a data center. In: Proc. of the 10th International Symposium on Pervasive Systems, Algorithms and Networks (2009)
[13] Wang, S., Bettati, R.: Reactive speed control in temperature-constrained real-time systems. In: Euromicro Conference on Real-Time Systems (2006)
[14] Xia, L., Zhu, Y.X., Yang, J., Ye, J., Gu, Z.H.: Implementing a thermal-aware scheduler in linux kernel on a multi-core processor. Computer Journal (2010)
[15] Zhang, S., Chatha, K.S.: Approximation algorithm for the temperature-aware scheduling problem. In: ICCAD (2007)
[16] Ryan, C., Sherief, R.: Thermal Prediction and Adaptive Control through Workload Phase Detection. ACM Transactions on Design Automation of Electronic Systems (TODAES) 18(1) (2013)
[17] Rahul, K., Jaiber, J., Thanunathan, R.: Phase-aware Predictive Thermal Modeling for Proactive Load-balancing of Compute Clusters. In: 2012 International Conference on Energy Aware Computing (2012)
[18] Hafiz, F.S., Hengxing, T., Ishfaq, A., Sanjay, R., Phanisekhar, B.: Energy- and Performance-aware Scheduling of Tasks on Parallel and Distributed Systems. ACM Journal on Emerging Technologies in Computing Systems (JETC) 8(4) (2012)
[19] Andrea, S., Paolo, C.: Cooling-aware Workload Placement with Performance Constraints. Performance Evaluation (2011)
[20] Marek, C., Christoph, D., Mathilde, H., Julien, R.: Algorithms for Temperature-aware Task Scheduling in Microprocessor Systems. Sustainable Computing: Informatics and Systems (2011)

An Improved Energy-Efficient Scheduling for Precedence Constrained Tasks in Multiprocessor Clusters[*]

Xin Li[**], Yanheng Zhao, Yibin Li, Lei Ju, and Zhiping Jia

School of Computer Science and Technology, Shandong University,
Jinan, 250101, China
lx@sdu.edu.cn

Abstract. Excessive energy consumption has become a critical issue in high performance computing. Task scheduling algorithms affect not only schedule length but also energy consumption. To shorten schedule length of parallel tasks with precedence constraints, scheduling algorithms could duplicate tasks on critical paths to avoid communication delay caused by inter-task dependence. However, task duplications incur more energy consumption. In this paper, we propose a heuristic Processor Reduction Optimizing (PRO) method to reduce the number of processors used to run parallel tasks, thereby decreasing system energy consumption. The PRO method can find appropriate time slots to accommodate tasks immigrated from low-utilized processors. The PRO method can be combined with existing duplication-based scheduling algorithms, such as Task Duplication Scheduling (TDS), Energy-Aware Duplication (EAD) scheduling and Performance-Energy Balanced Duplication (PEBD) scheduling. Experimental results show that the proposed PRO method can effectively decrease the number of used processors and save energy while maintaining schedule length.

Keywords: cluster, energy consumption, DAG, task dependency.

1 Introduction

Computer clusters bring high performance as well as large energy consumption. For example, the Environment Protection Agency reported that the total energy consumption of servers and data centers in the United States in 2006 was 61.4 billion KWh [1], which was almost equal to the total energy cost of 5.8 million US households. Energy conserving techniques appear to be important for high performance computing.

Task scheduling problems on multiple processors have been proved to be NP-hard [2]. Many heuristic scheduling algorithms have been proposed to reduce schedule length (i.e., makespan) or energy consumption. For example, task duplication scheduling [3]

[*] An abstract containing some preliminary results of this paper appeared in the Chinese Journal of Computer, 2012, 35(3): 591-602. This paper is an extended English version based on the same study.
[**] Corresponding author.

eliminated communication delay between processors by running redundant critical tasks on the processors. Most duplication-based scheduling algorithms replicate all possible tasks to shorten schedule length, however, neglecting energy consumption caused by task duplications. Zong *et al.* [4] proposed two non-preemptive off-line energy-saving scheduling algorithms—EAD and PEBD to balance schedule length and energy savings by judiciously duplicating predecessors of a task if the duplication can aid in performance without degrading energy efficiency. These algorithms assume that unbounded resources are available in a cluster. However, the number of processors is limited in real clusters. In addition, these scheduling algorithms do not consider the issue of unbalanced workloads. Fig. 1 shows the processor utilizations for a real word application FPPPP [5] scheduled by TDS, EAD and PEBD algorithms. The execution time of the first 10% cores takes 33.6%, 48.9% and 38.6% of the total execution time of all 222 processors, and most processors stay in the idle mode on 95%, 96.9% and 95.9% of all time, respectively.

In this paper, we propose a heuristic Processor Reduction Optimizing (PRO) method to find appropriate time slots to re-group tasks. A group of tasks will be assigned to a processor to execute. The PRO approach can merge low-utilized processors and reduce the number of processors used to run parallel tasks, thereby decreasing system energy consumption. Combining the PRO method with three existing algorithms—TDS, EAD and PEBD, we propose three improved algorithms—TDS-PRO, EAD-PRO and PEBD-PRO. Experimental results show that the schedule lengths of the improved algorithms are equal to or slightly less than those of the original algorithms. However, TDS-PRO, EAD-PRO and PEBD-PRO can conserve energy by 23.7%, 21.3% and 23.5% and decrease the number of processors by 21.2%, 22.8% and 20.9%, respectively.

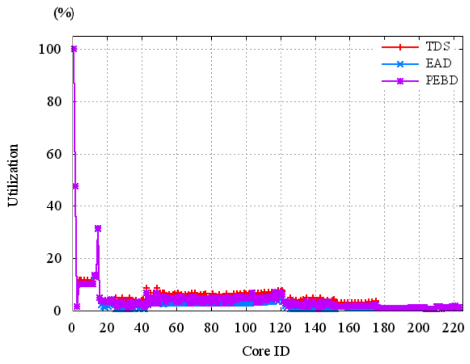


Fig. 1. Utilization under original algorithms

The rest of the paper is organized as follows. In section 2, we present related work. In section 3, we introduce mathematical models including a task model, a cluster model, a dependency model and an energy consumption model. In section 4, we present the improved task scheduling strategy. Experimental environment and simulation results are demonstrated in section 5. Finally, section 6 provides the conclusions.

2 Related Work

Generally, parallel scheduling strategies can be classified into three primary categories, namely priority-based scheduling, cluster-based scheduling, and duplication-based scheduling [4]. Priority-based scheduling [6] assigns a priority level for each task, and maps tasks to processors according to assigned priorities. Cluster-based scheduling [7] allocates a group of tasks which have much intercommunication to one processor, thereby dropping communication overheads. Duplication-based scheduling [3] replicates some common predecessor tasks in critical paths to shorten schedule length. In most cases, the performance of duplication-based scheduling is superior to non-duplication scheduling, especially when communication time dominates the execution time of parallel applications. However, executing multiple copies of replicated tasks on different processors may increase energy consumption of cluster systems if execution energy is greater than saved communication energy. To solve this problem, Zong *et al.* [4] proposed two energy-efficient duplication-based algorithms, in which Energy-Aware Duplication (EAD) scheduling set an energy increment threshold for judging whether critical tasks should be replicated or not, while Performance-Energy Balanced Duplication (PEBD) scheduling set a cost ratio of energy increment to reduced schedule length as the threshold. To improve the EAD and PEBD, our method moves tasks on low utilized processors to heavy or medium utilized processors with appropriate time slots for these moved tasks. Our method reduces the number of processors, thereby reducing energy consumption. This paper is similar to our past publication in [8] that was written in Chinese. This is an extended version based on the same work.

The duplication-based algorithms mentioned above get a high schedule performance on unbounded processors, but it is unreasonable in real world. Gerasoulis *et al.* [8] proposed a multi-stage mechanism to schedule parallel tasks to a multiprocessor system with a bounded number of processors. In [9], a reverse clustering approach tried to find the minimum parallel time for DAGs on a system with a given number of processors. Sarkar [10] used a clustering procedure as its first step, with the assumption of unlimited number of completely connected processors, and then the task clusters were merged and scheduled on a multiple processor system. Two list scheduling heuristics [11] used critical path information and ready list priority scheduling to group tasks. Our method can be combined with these algorithms to decrease the number of used processors.

3 Mathematical Model

In this section, we describe mathematical models used to represent homogeneous clusters, precedence-constrained parallel tasks and energy consumption.

3.1 Task Model

Parallel applications with precedence-constrained tasks can be represented in form of a Directed Acyclic Graph (DAG) [12]. A parallel application running in clusters is modeled as a vector pair (*V*, *E*), where $V = \{v_i | 1 \leq i \leq n\}$ denotes a set of *n* parallel tasks, and $E= \{e_{ij}| v_i,v_j \in V\}$ represents a set of message communications and precedence constraints among tasks. It is assumed that every task is a non-preemptive and indivisible work unit. The execution time of task v_i is represented as t_i. We assume that there is only one entry task and one exit task in a DAG. The assumption is reasonable because in case of multiple entry or exit tasks existing, a dummy entry task or exit task will be added. Fig. 2(a) shows a DAG example of 5 tasks. Task 6 is a dummy exit task. The number in a vertex is task number. The value around a vertex represents task execution time t. Value c_{ij} on edge e_{ij} indicates the communication time of passing the messages e_{ij} if the task i and j are not executed on the same processor. The cost c_{ij} is set to zero if the task i and j run on the same processor.

A task execution path l (termed as path for short) includes one or more sequential tasks with linear precedence. All tasks on one path must be executed under precedence constraints. The path with the largest sum of task execution time is called a critical path. A critical path has an entry task and an exit task. In Fig. 2 (a), the path $<v_1, v_4, v_5>$ is the only critical path, in which task v_4 must be executed after v_1 and before v_5. All paths are generated by a duplication-based algorithm that will be discussed in the subsection 4.3.

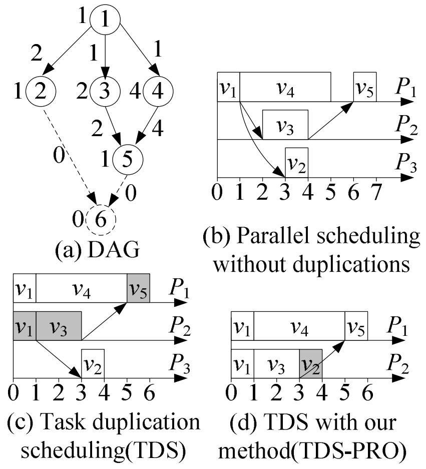

(a) DAG
(b) Parallel scheduling without duplications
(c) Task duplication scheduling(TDS)
(d) TDS with our method(TDS-PRO)
P1
P2
P3
v1
v2
v3
v4
v5

Fig. 2. An example of parallel task scheduling

3.2 Cluster Model

A cluster in this study is characterized by a set $\{P_1, P_2,..., P_M\}$ of homogeneous computational nodes connected by high-speed interconnects. It is assumed that each node has a single core processor and its frequency is fixed.

Task allocation matrixes X and Y are defined as below:

$$X = \{x_{ij}|1\leq i \leq n,\ 1\leq j \leq M\},$$

in which, x_{ij} is set to 1 if task i is assigned to processor j; otherwise, x_{ij} is set to 0. One execution of task i running at processor j is called a job w_{ij}. All executions of task i can be denoted as $\{w_{ij}| x_{ij}=1, 1\leq j \leq M\}$.

Fig. 2(b)-(d) show schedule results using three schedule algorithms, in which arrows denote the message communications if two precedence-constrained tasks do not run on a same processor. The details will be discussed in section 4.

3.3 Energy Consumption Model

The processor is the most energy-intensive part of a cluster system [4]. Energy consumption of switches accounts for 80% of the total energy consumption in network equipments [13]. Therefore, in this study, we only count energy cost of processors, network cards and switches as the system energy consumption.

3.3.1 Processor Energy

We assume that each processor core has two operating modes: a busy mode and an idle mode. The processor power in the busy mode is denoted as PC_{busy} and the power in the idle mode is PC_{idle}. One processor will cost some energy to run a task. We use Ev_i to denote the energy cost by a processor to run task v_i.

$$Ev_i = PC_{busy} \times t_i \tag{1}$$

Since multiple jobs of one task may run on different processors to shorten schedule length, the energy of all processors consumed in the busy mode (EC_{busy}) is computed as:

$$EC_{busy} = PC_{busy} \sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij} \cdot t_i \tag{2}$$

The energy of processor j in its idle mode is equal to

$$EC_{idle}^{j} = PC_{idle} \left(L_{max} - \sum_{i=1}^{n} (x_{ij} \cdot t_i) \right) \tag{3}$$

in which $\sum_{i=1}^{n} (x_{ij} \cdot t_i)$ is the sum of execution time of tasks executed on processor j, and L_{max} is the schedule length, i.e., the completion time of the last task.

$$L_{max} = \max_{i=1}^{n} \max_{j=1}^{m} (f_{ij}) \tag{4}$$

in which f_{ij} depicts the completion time of job w_{ij}.

The energy consumed by all processors in their idle modes (EC_{idle}) is equal to,

$$EC_{idle} = PC_{idle} \left(m \cdot L_{max} - \sum_{j=1}^{m} \sum_{i=1}^{n} (x_{ij} \cdot t_i) \right) \tag{5}$$

The total energy used by processors is defined as:

$$EC = EC_{busy} + EC_{idle} \tag{6}$$

This energy consumption model is compatible with the DVFS technology, in which processors may have several voltage and frequency levels. Scheduling algorithms may choose the best fit frequency to conserve energy. In that case, PC_{busy} can be replaced with $PC_{best\text{-}fit}$ in the model.

3.3.2 Network Energy

Energy consumption of network interconnections is expressed as *EL=EN+ES*, in which *EN* denotes the energy consumption of network interface cards, and *ES* represents the energy consumption of switches.

$$EN = M \cdot PN_{idle} \cdot L_{\max} + 2 \cdot (PN_{busy} - PN_{idle}) \cdot \sum_{i=1}^{n} \sum_{j=1}^{M} (x_{ij} \cdot \sum_{e_{ki} \in E} (1 - x_{kj}) \cdot c_{ki}) \quad . (7)$$

PN_{busy} and PN_{idle} represent the busy and idle power of network cards, respectively. x_{ij} is set to 1 if v_i is assigned to processor *j*; otherwise, x_{ij} is 0. c_{kj} is the communication time from task *k* to task *j*. We calculate the energy consumed by a sender and a receiver, so the second term in Eq.(7) includes a multiplier of two.

In [14], it is indicated that a network interconnection equipment (e.g., switch) has the same power in the idle mode and the busy mode, so we set a unified parameter *PS* as the power of one switch. Energy consumption of switches (*ES*) is defined as following:

$$ES = N_{switch} \cdot L_{\max} \cdot PS \tag{8}$$

where N_{switch} denotes the number of switches. It is assumed that all processors are connected to a two-layer cascade switch network. N_{port} represents the number of ports of every switch. N_{switch} is decided by two parameters: *M* and N_{port}.

$$N_{switch} = \begin{cases} 1 \quad , & M \leq N_{port} \\ \left\lceil \dfrac{M}{N_{port}} \right\rceil + 1, & N_{port} < M \leq (N_{port})^2 \end{cases} \tag{9}$$

The energy consumption of a homogeneous cluster can be expressed as:

$$E = EC + EN + ES \tag{10}$$

4 Task Assignment and Optimization

In this section, we present a duplication-based task assignment and optimization procedure to create task execution paths and re-group the paths to reduce the processor count. The entire procedure consists of five steps delineated in the subsection 4.1-4.5.

4.1 Generate Original Task Scheduling Sequence

Precedence constraints of DAG tasks have to be guaranteed by executing predecessor tasks before successor tasks. To achieve this goal, the first step is to generate an ordered task sequence using the concept of level. The level of a task is defined as the sum of computation time from the task to the exit task. We use a bottom-up approach proposed in [3] to define the level of v_i.

$$Level(v_i) = \begin{cases} t_i, & Succ(i) = \Phi \\ \max\limits_{e_{ik} \in E}\{Level(v_k)\} + t_i, & \text{otherwise} \end{cases} \tag{11}$$

The *Succ*(*i*) denotes as a set of successor tasks after task *i*. The level of an exit task is its execution time and the levels of other tasks can be calculated by recursively applying the second term in Eq. (11). All tasks are sorted in an ascending order of their levels and the sorted tasks form the original task scheduling sequence.

4.2 Parameter Calculation

The second step is to calculate task schedule parameters, which are used to make duplication decisions by the path generating algorithm (discussed in the subsection 4.3). The important parameters include the Earliest Start Time(*EST*), the Earliest Completion Time (*ECT*), the Latest Allowable Start Time (*LAST*), the Latest Allowable Completion Time (*LACT*) and the Favorite Predecessor (*FP*). The similar parameters were first proposed in [15].

The *EST* of a task can be calculated in a top-down manner by Eq. (12):

$$EST(v_i)=\begin{cases} 0, & Pred(i)=\Phi, \\ \min\limits_{e_{ji}\in E}\left(\max\limits_{e_{ki}\in E, v_k\neq v_j}\begin{pmatrix} ECT(v_j), \\ ECT(v_k)+c_{ki}\end{pmatrix}\right), & \text{otherwise}\end{cases} \tag{12}$$

in which, the *Pred*(*i*) denotes as a set of predecessor tasks before task *i*.

The Earliest Completion Time (*ECT*) of task v_i is expressed as the sum of its earliest start time and execution time. Thus, we have:

$$ECT(v_i)=EST(v_i)+t_i \tag{13}$$

The Favorite Predecessor (*FP*) of v_i is one of predecessors of v_i which will pass the message to v_i at latest if they are not on a same processor. $FP(v_i)$ is defined as below:

$$FP(v_i)=v_j,\quad where\,\forall e_{ji}\in E, e_{ki}\in E, j\neq k \mid ECT(v_j)+c_{ji}\geq ECT(v_k)+c_{ki} \tag{14}$$

The Latest Allowable Completion Time (*LACT*) of a task is calculated in a top-down manner by recursively applying the second term of Eq. (15).

$$LACT(v_i)=\begin{cases} ECT(v_i), & Succ(i)=\Phi \\ \min\left\{\min\limits_{e_{ij}\in E, v_i\neq FP(v_j)}\{LAST(v_j)-c_{ij}\},\ \min\limits_{e_{ij}\in E, v_i=FP(v_j)}\{LAST(v_j)\}\right\}, & otherwise\end{cases} \tag{15}$$

The Latest Allowable Start Time (*LAST*) of task v_i is derived from its latest allowable completion time and execution time.

$$LAST(v_i)=LACT(v_i)-t_i \tag{16}$$

4.3 Generate Initial Execution Paths

After obtaining the original task sequence and the important parameters, we use a duplication-based scheduler (TDS[3], EAD[4] or PEBD [4]) as the path generating algorithm to produce the initial execution paths. The objective of this step is to assign task v_i and its *FP* to an execution path to reduce the communication cost. For the example in Fig. 2, TDS will create three initial paths, i.e., l_1=<v_1, v_4, v_5>, l_2=<v_1, v_3>, and l_3=<v_2>. Task v_1 will be executed for two times, because its replications can reduce the schedule length from 7 to 6.

4.4 Update Task Schedule Time

```
UpdateTaskScheduleTime($v_i$)
1.  for each $w_{ij} \in v_i$
2.    if $Pred(i)=\Phi$ then
3.      $EST(w_{ij})=0$
4.    else then
5.      $startTime= -1$ //lower boundary
6.      for $v_k \in Pred(i)$
7.        if $x_{kj}=1$ then
8.          $preECT=ECT(w_{kj})$
9.        else then
10.         $temp1=+\infty$ // Upper boundary
11.         for $P_l \in \{P_j | x_{kj}=1\}$
12.           if $ECT(w_{kl})$ is unknown then
13.               $UpdateTaskScheduleTime(v_k)$
14.           $temp2=ECT(w_{kl})+c_{ki}$
15.           if $temp2 <= temp1$ then
16.             $temp1=temp2$
17.         $preECT = temp1$
18.       if $preECT > startTime$ then
19.         $startTime=preECT$
20.     if $w_{ij}$ is not the first task on $P_j$ then
21.       $z$=the job just before $w_{ij}$
22.       if $ECT(z)$ is unknown then
23.         $UpdateTaskScheduleTime(z)$
24.       if $ECT(z)>startTime$ then
25.         $startTime=ECT(z)$
26.     $EST(w_{ij})=startTime$
27.     $ECT(w_{ij})=startTime+t_i$
```

Fig. 3. Pseudo code of the function to update task schedule time

Because a task may run more than once at different processors in a duplication-based scheduler, the *EST* and *ECT* of every task should be updated. We traverse each task of the DAG in a breadth first manner to calculate its parameters. If task *i* is a entry task, i.e. $Pred(i)=\Phi$, its *EST* is equal to 0; otherwise, the EST_{ij} is equal to the latest one of the arrival time (i.e. completion time + communication time) of all processors and the completion time of the task before task *i* on processor *j*. The pseudo code of the update function is given in Fig. 3. The local variable *startTime* is used to store the temp value of the EST during the function runs.

4.5 Processor Reduction Optimization

Compared with TDS, EAD and PEBD reduce low energy-efficient task duplication to decrease energy consumption. However these three algorithms ignore low utilized processors which waste a lot of energy, especially when tasks have many execution paths.

To reduce more energy, we propose a processor reduction optimization (PRO), which moves tasks in low utilized processors to other processors. This method tries to use fewer processors to run all tasks while maintaining the same or less schedule length.

The main idea beside the PRO method is to find appropriate slack time slots in medium or heavy utilized processors to accommodate tasks assigned to low utilized processors. Thus the low utilized processors will not be used and the energy will be conserved.

The PRO method is shown in the following:

```
ProcessorReductionOptimizing()
1.  sort the processor sequence $\{P_1,P_2,...,P_M\}$ in a descending order of $|P_i|$,
    the number of jobs in $P_i$.
2.  for $i \leftarrow M$ to 2// preferably optimize less utilized processors
3.      if $P_i$ has immigrated tasks then
4.          continue
5.      for $k \leftarrow |P_i|$ to 1//try to move tasks in $P_i$.
6.          $w$=the $k$-th job on $P_i$
7.          bTag=false
8.          for $j \leftarrow 1$ to $i$-1 //try to find an appropriate slot
9.              if TryMoveJob($w$, $P_j$,$P_i$) then
10.                 bTag=true
11.                 break
12.         if bTag=false then
13.             break//if one task cannot be moved, stop optimizing $P_i$
```

Fig. 4. Pseudo code of the processor reduction optimizing method

The *TryMoveJob* function (line 9) is an important part in the PRO method. The *TryMoveJob* function tries to move the job w from P_i into the slack slots in P_j. We assume the job w is a copy of task i. Since a task may run multiple copies on different processors in duplicated scheduling, the function checks whether the task i is assigned to processors except P_i. If so, it is not necessary to execute the job w again on P_j. The function will return true and the job w will be deleted directly from P_i. If the job w is the only copy of the task i, the function will try to assign to job w an appropriate time slot on P_j. We show the function in Fig. 5.

The PRO method can be used to improve the performance of TDS, EAD and PEBD. The improved scheduling are termed TDS-PRO, EAD-PRO and PEBD-PRO. For example, Fig. 2 (b)-(d) show the results of three scheduling algorithms, including parallel scheduling without duplications, task duplication scheduling (TDS) and TDS with our method (TDS-PRO). Parallel scheduling without duplications allows v_2, v_3, v_4 and v_5 to run at the same time on three different processors. In Fig. 2(d), TDS shortens the schedule length, as it executes redundant tasks v_1 on the processor P1 and P2, and avoids the message passing time from task v_1 to v_3. TDS saves communication cost between processors, while it may increase energy consumption of the system if the execution energy of the redundant task is greater than the reduced communication energy. In Fig. 2(d), the improved scheduling TDS-PRO moves the third execution path $<v_2>$ to processor P_2. The PRO method reduces the number of used processors under the constraint of no increment in the schedule length, thus the method can save more energy.

```
TryMoveJob($w$, $P_{to}$, $P_{from}$)
1.  if $w$ runs at one or more times in other processors then
2.      delete $w$ from $P_{from}$
3.      return true
4.  for $preTask \in Pred(w)$
5.      if $P_{to}$ has no preTask then
6.          if $P_{from}$ has preTask then
7.              return false //if $w$ is delayed, do not move $w$
8.  for $i \leftarrow 0$ to $|P_{to}|$
9.      if $i$=0 then
10.         STslot=0//start time of the slot
11.         ETslot=EST(the first job in $P_{to}$)
12.     else if $i=|P_{to}|$ then//in the end of job queue
13.         STslot=ECT(the last job in $P_{to}$)
14.         ETslot=$L_{max}$ //not greater than the schedule length
15.     else
16.         STslot= ECT(the $i$-the job in $P_{to}$)
17.         ETslot =EST(the $i$-1 the job in $P_{to}$)
18.     if $STslot \leq EST(w)$ and $ECT(w) \leq ETslot$ then
19.         delete $w$ in $P_{from}$
20.         move $w$ to $P_{to}$
21.         return true
22.     else return false
```

Fig. 5. Pseudo code of the *TryMoveJob*() function

4.6 Time Complexity Analysis

In this subsection, we analyze the time complexity of the whole procedure of task assignment and optimization.

Theorem 1. Given a parallel application with multiple precedence-constrained tasks, the time complexity of the PRO scheduling is $O(h^2n^2)$, where n is the number of parallel tasks, h is the height of the DAG.

Proof. The whole strategy performs five steps, described in subsection 4.1-4.5. In the first step, all tasks of the DAG are traversed to compute their levels. And then, some important parameters like *EST*, *ECT*, *FP*, *LACT*, and *LAST* are calculated. In the third step, all tasks are assigned to one or more initial task paths by a duplication-based algorithm. The overall time complexity of the first three steps is $O(2|E|+n(\lg n+1)+hn)$ [4], in which $|E|$ is the number of edges in the DAG. The fourth step is performed to update task schedule time. In the worst case, all jobs will be visited. Its time complexity is $O(hn(|E|+n))$. In the last step, the sort complexity is $O(n\lg n)$ and the *TryMoveJob* function has a complexity of $O(hn)$. The function is called for hn times at most. Thus the complexity in the last step is $O(h^2n^2)$.

Consequently, the overall time complexity is the sum of those in all steps, i.e. $O(2|E|+n(\lg n+1)+hn+ hn(|E|+n)+ h^2n^2)$. Since $|E|<hn$, the complexity is $O(h^2n^2)$.

5 Performance Evaluation

In order to evaluate the performance of the proposed method, we developed a software simulator using C++ and implemented three baseline algorithms and three improved algorithms. In this section, we give experiment settings and results.

5.1 Experiment Settings

5.1.1 Hardware Configuration

We assume that computers are connected by Gigabit Ethernet and switch power is equal to 75W and NIC power is 5W. We use four models of processors from AMD and Intel corporations in the experiment. Table 1 shows busy power and idle power [23] of every processor.

Table 1. Parameters of CPUs

Processor	Busy power	Idle power
AMD Athlon 4600+ 85W	104W	15W
AMD Athlon 4600+ 65W	75W	14W
AMD Athlon 4600+ 35W	47W	11W
Intel Core 2 Duo E6300	44W	26W

5.1.2 Task Sets

In the experiment, we simulate two real-world parallel applications—the Robot Control application (with 88 tasks and 131 edges) and the FPPPP application (with 334 tasks and 1,145 edges). The detailed parameters of the task sets from [5] are listed in Table 2.

Since there are precedence constraints among tasks in these two task sets, a Communication-Computation-Ratio (*CCR*) is used to create communication costs. The communication cost from v_i to v_j is calculated by following:

$$c_{ij} = CCR \cdot t_i \tag{17}$$

Table 2. Two task sets used in the experiments

Task set	Task count	Edge count	Avg. out-degree	Avg. execution time	EAD_Th	PEBD_Th
Robot	88	131	1.5	28.2	200	80
FPPPP	334	1145	3.6	21.3	1500	150

CCR may change when an application runs on different processors and interconnects. Thus, we varied *CCRs* in a reasonable range of 0.1 to 2. All tunable parameters in our experiments are listed in Table 3.

Table 3. Tunable Parameters used in the experiments

No	Processor	Task Set	*CCR*
Test 1	AMD 35W	Robot/FPPPP	0.1
Test 2	AMD85W/65W/35W/E6300	Robot	0.1
Test 3	AMD 35W	Robot	0.1-2

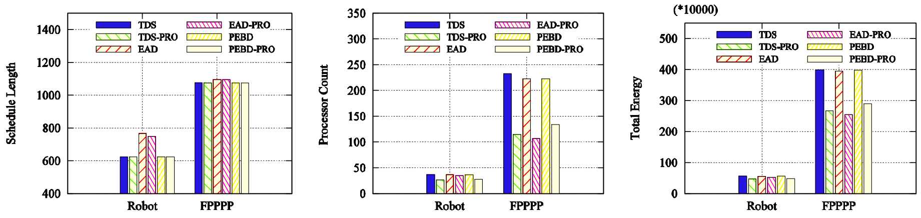
5.1.3 Baseline Algorithms

Three duplication-based algorithms, i.e., TDS [4], EAD [7] and PEBD[7], are tested as the baseline algorithms. The goal of TDS is to shorten schedule lengths, whereas EAD and PEBD aim to achieve tradeoff between schedule length and energy consumption. These algorithms allocate all tasks in an execution path to one processor.

The thresholds (EAD_Th and PEBD_Th) used in the experiments are listed in Table 2. Please note that we set different thresholds for the two task sets to achieve their best performance.

5.2 Test 1: Impact of Different Task Sets

In test 1, the Robot task set and FPPPP task set are used to evaluate the impact of different task sets on algorithms on in the experiment. The result is shown in Fig. 6.



(a) schedule length (b) processor count (c) total energy

Fig. 6. Impact of task sets

Figure 6(a) shows the schedule length of all the algorithms. Because the PRO method moves a task only when its successor task will be delayed, three improved algorithms have similar schedule length to the baseline algorithms. Figure 6(b) shows the processor count required by the Robot and FPPPP task set. On the Robot task set, TDS-PRO, EAD-PRO, PEBD-PRO decrease the processor count by 27%, 5.4% and 24% in comparison with TDS, EAD and PEBD respectively, while on the FPPPP task set they decrease 51%, 52% and 40% over their baseline algorithms. It can be found that the task set with more paths (such as FPPPP) requires more processors. Processors have lower processor utilizations on FPPPP, so that more processors can be saved.

5.3 Test 2: Impact of Different Processor Models

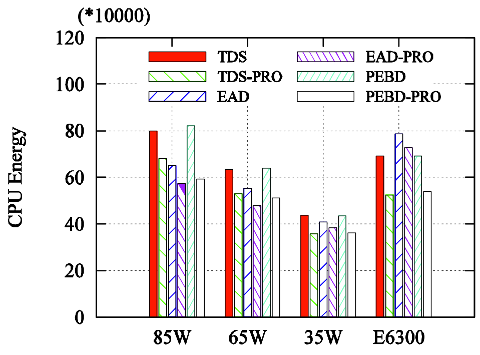


Fig. 7. Total energy consumption of different CPUs

Fig. 7 shows that, TDS-PRO, EAD-PRO and PEBD-PRO consume less energy than their corresponding baseline algorithms. TDS-PRO can save 15%, 16.5%, 18% and 24% energy over TDS on four processor models respectively, while EAD-PRO can save 12%, 13%, 6% and 7% energy over EAD and PEBD-PRO can save 28%, 20%, 17% and 22% energy over PEBD.

5.4 Test 3: Impact of Different Communication-Computation-Ratios

This test shows the impact of different CCRs. The value of CCR changes from 0 to 2, stepping 0.1.

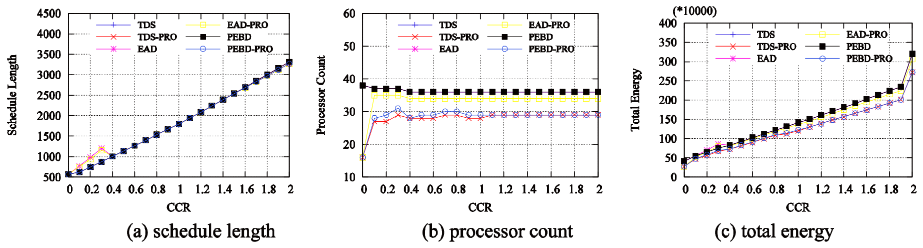


Fig. 8. Impact of different CCRs

When the CCR changes from 0.4 to 2 gradually in Fig. 8(a), the schedule length of all the algorithms increases as well. It is because higher CCR causes higher network latency, and thus the schedule length increases. Meanwhile, the total energy consumption increases for higher transmission energy consumption.

When CCR varies from 0 to 0.3 in Fig. 8(b), the number of processors required in TDS, EAD and PEBD is slightly reduced. And the PRO algorithms reduce the number of processors obviously, which shows that CCR not only affects the network communication time between processors, but also affects the optimization effect of the PRO algorithms.

In Fig. 8(c), the total energy consumption of all the algorithms grows with the increasing CCR. And the PRO algorithm performs slightly better than the baseline algorithm.

5.5 Experiment Summary

According to the three tests above, we know that the improved algorithms can reduce the number of used processors and energy consumption greatly while the schedule length is the same or less than that of original algorithms. The TDS-PRO, EAD-PRO and PEBD-PRO decrease average energy consumption by 23.7%, 21.3% and 23.5% and reduce average number of processors by 21.2%, 22.8% and 20.9%, respectively.

6 Conclusions

In this paper, we formalized the energy-efficient task scheduling problem on homogeneous clusters. A duplication-based algorithm was employed to create initial execution paths. Then we proposed a heuristic Processor Reduction Optimizing (PRO) method to reduce the number of processors used to run parallel tasks, thereby decreasing system energy consumption. The PRO method can find appropriate time slots to accommodate tasks immigrated from low-utilized processors. Compared with three existing algorithms, TDS, EAD and PEBD, the improved algorithms can significantly reduce energy consumption for homogeneous clusters, especially for communication-intensive tasks sets.

Acknowledgements. This research is sponsored by the Natural Science Foundation of China (NSFC) under grants No. 60903031, 61070022, 61202015, National 863 Program 2013AA013202, Research Fund for the Doctoral Program of Higher Education of China (RFDP) grant 20120131120033, and the Outstanding Young Scientist Foundation of Shandong Province (BS2010DX017).

References

1. http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf
2. Huang, J.G., Chen, J.E., Chen, S.Q.: Parallel-job scheduling on cluster computing systems. Chinese Journal of Computer 27(6), 765–771 (2004)
3. Ranaweera, S., Agrawal, D.P.: A task duplication based scheduling algorithm for heterogeneous systems. In: The Parallel and Distributed Processing Symposium, Atlanta, USA, pp. 445–450 (2000)
4. Zong, Z.L., Manzanares, A., Ruan, X.J., Qin, X.: EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters. IEEE Transactions on Computers 60(3), 360–374 (2011)
5. Standard Task Graph Set web site (2011): http://www.kasahara.elec.waseda.ac.jp/schedule

6. Sih, G.C., Lee, E.A.: A compile time scheduling heuristic for interconnection-constrained heterogeneous processors architectures. IEEE Transactions on Parallel and Distributed Systems 4(2), 175–187 (1993)
7. Pande, S.S., Agrawal, D.P., Mauney, J.: A scalable scheduling method for functional parallelism on distributed memory multiprocessors. IEEE Transactions on Parallel and Distributed Systems 6(4), 388–399 (1995)
8. Li, X., Jia, Z.P., Ju, L., Zhao, Y.H., Zong, Z.L.: Energy Efficient Scheduling and Optimization for Parallel Tasks on Homogeneous Clusters. Chinese Journal of Computer 35(3), 591–602 (2012) (in Chinese)
9. Gerasoulis, A., Yang, T.: Scheduling program task graphs on MIMD architectures. Parallel Algorithm Derivation and Program Transformation, 153–186 (1993)
10. Zhou, H.B.: Scheduling DAGs on a Bounded number of Processors. In: International Conference of Parallel and Distributed Processing Techniques and Applications, pp. 823–834 (1996)
11. Sarkar, V.: Partitioning and scheduling parallel programs for multiprocessors. The MIT Press (1989)
12. Yang, T., Gerasoulis, A.: List scheduling with and without communication delays. Parallel Computing 19, 1321–1344 (1993)
13. Kwok, Y.K., Ahmad, I.: Efficient Scheduling of Arbitrary Task Graphs to Multiprocessors Using a Parallel Genetic Algorithm. J. Parallel and Distributed Computing 47(1), 58–77 (1997)
14. Gupta, M., Singh, S.: Greening of the Internet. In: The ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications (SIGCOMM 2003), Karlsrhue, Germany, pp. 19–26 (2003)
15. Hsu, C.H., Feng, W.C.: A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters. In: IEEE Cluster Computing (Cluster 2005), Burlington, USA, pp. 1–10 (2005)
16. Darbha, S., Agrawal, D.P.: A Task Duplication Based Scalable Scheduling Algorithm for Distributed Memory Systems. J. Parallel and Distr. Comp. 46(1), 15–27 (1997)
17. `http://www.xbitlabs.com/articles/cpu/display/amd-energy-efficient_6.html`

Hierarchical Eventual Leader Election for Dynamic Systems

Huaguan Li[1], Weigang Wu[2], and Yu Zhou[3]

[1] Department of Computer Science,
Sun Yat-sen University,
Guangzhou 510006, China
wuweig@mail.sysu.edu.cn
[2] College of Computer Science,
Nanjing University of Aeronautics and Astronautics
Nanjing 210016, China
zhouyu@nuaa.edu.cn

Abstract. Dynamic system is a recent hot research topic in theoretical distributed computing. The dynamicity caused by process join and leave bring new challenges in fundamental distributed computing problems, such as eventual leader election. In this paper, we consider leader election in dynamic systems with cluster-based hierarchy. Clustering based hierarchy has been used in fundamental distributed algorithms to achieve scalability and low communication cost, but, to the best of our knowledge, it is not considered in eventual leader election, especially in eventual leader for dynamic systems. We firstly define new system models to describe the dynamicity of clusters, and then based on these models, we design an algorithm to elect an eventual leader. With cluster hierarchy, leader election is basically conducted in two layers. In the lower layer, cluster-heads are elected with each cluster. Then, in the upper layer, election is conducted among cluster-heads so as to elect the eventual leader of the whole system. Several key challenging issues caused by cluster dynamicity have been addressed in our design, including blocking in election within a cluster and multiple cluster-heads in election of upper layer. The proposed algorithm is proved to be correct rigorously.

Keywords: Eventual leader, consensus, distributed algorithms, dynamic systems.

1 Introduction

In the recent years, dynamic system [11] is attracting more and more attentions from researchers in distributed computing. Dynamic system[1] refers to a distributed system with processes entering and leaving the system dynamically. Dynamic system is in fact

[1] Please notice that “dynamic network" focuses on the topology change due to node mobility and sometimes is also called “dynamic system". However, in this paper, “dynamic system" refers to system with changing process set.

the abstraction of many instance systems, including peer-2-peer systems [6][7] and wireless ad hoc network systems [8]. Study of dynamic systems provides theoretical support for the development of these instance systems.

The research on distributed system mainly includes two aspects. Dynamic system models [11][12] focus on the definition and assumptions about the behaviors of a dynamic system, especially the dynamicity of processes. System model is the basis of other distributed computing problems. More efforts on dynamic systems have been made to design algorithms for fundamental distributed problems, e.g. information dissemination [9][10].

Eventual leader election[4] is also a fundamental problem in distributed computing. An eventual leader in dynamic systems can be generally defined as [11]:

Definition 1. Eventual leader: Eventually all the stable processes are provided with a single leader that is a stable process. The meaning of "stable" will be defined later.

Eventual leader election can be used as an oracle to solve coordination problems such as consensus [21]. People have done much work to eventual leader election, and many good algorithms have been proposed for static distributed systems [1][2][3][15]. Recently, eventual leader in dynamic systems has also attracted much attention from researchers, and several algorithms have been proposed [4][5][11].

In this paper, we consider cluster based hierarchy in dynamic leader election. Clustering based hierarchy has been used in fundamental distributed algorithms, including consensus [23][24] and information dissemination [18], to achieve scalability and low communication cost, but, to the best of our knowledge, it is not considered in eventual leader election, especially in eventual leader for dynamic systems.

With the help of cluster hierarchy, election of an eventual leader can be realized in two layers. In the lower layer, election is conducted to select cluster-head inside a cluster. In the upper layer, election is conducted among cluster-heads to select the globally unique leader. With such a hierarchical design, the cost of leader election will be significantly reduced.

Intuitively, hierarchical election can be achieved by combining two election algorithms in the lower layer and upper layer respectively. However, due to the dynamicity of the cluster hierarchy, it is far from trivial to combine the elections in two layers, and existing eventual leader algorithms, either dynamic or static, cannot be directly adopted. More precisely, there are three key challenges to be addressed to realize a hierarchical election in dynamic systems.

Firstly, the dynamicity models for general dynamic systems is not feasible in a cluster, because a cluster is only part of the whole system. This may cause the election in the lower layer blocked forever.

Secondly, due to the eventual property of the leader, the cluster-head set may change from time to time and there may be multiple cluster-heads existing in a single cluster during algorithm execution. This makes the upper layer also a dynamic set of part of the whole system.

Thirdly, since only cluster-heads participate in the upper layer election, the member process of a cluster may not be able to get the result of the upper layer.

Especially, how those processes blocked in a cluster with not enough processes can obtain the leader information is not easy.

Our design is conducted with respect to the challenges described above. We firstly propose models/assumptions to define the dynamicity of clusters and processes in a cluster, and then design algorithms to elect an eventual leader. Our work in this paper is based on the models and algorithms in [11] and [15]. More precisely, we extend the α model to be workable both inside a cluster and among cluster-heads; we modify the algorithms in [11][15] and design new additional mechanism to address the challenges aforementioned.

The rest of the paper is organized as follows. Section 2 reviews existing works on the eventual leader election, especially those for dynamic system. We describe system models and assumptions on dynamicity in Section 3, and present the hierarchical algorithm in Section 4. The correctness proof is provided in Section 5. Finally, Section 6 concludes the paper with further directions.

2 Related Work

Most of existing eventual leader election algorithms are proposed for static systems [1][2][3][13][15][20], where the set of processes is fixed and known.

Time based algorithms differ in the degree of synchrony. The first approach investigated in [1][2] considers that all the links connecting the processes are eventually timely. This means that after some time τ, each message reaches its destination in at most δ units of time. This approach has been refined to obtain weaker constraints. It has been shown in [16] that Ω can be implemented in a system where at least one correct process has at least s eventually timely outgoing links (this is defined as s-source).

Interestingly, a step ahead has been taken in [20], where the notion of eventual s-accessibility is introduced. Informally, a process p is s-accessible at some time if messages sent by p at that time are received within δ units of time by a set Q of at least s processes. The interest of this notion lies in the fact that the set Q of processes that “witness” p can be different at different time.

A time-free approach [15][19] implements the eventual leadership using a query-response based mechanism. There are totally n processes in the system and at most t of them can fail (by crashing only). The solutions in [15][19] rely on an assumption on the behavior of the flow of message exchange. More precisely, processes broadcast queries and then wait for responses from other processes. The first $n\text{-}t$ responses received are winning responses (the other responses, if any, are called losing responses; they can be slow or never sent due to the crash of the sender never sent due to the crash of the sender). It is shown in [15] that Ω can be built if the following behavioral property is satisfied: "there is a correct process p and a set Q of $t+1$ processes such that eventually each response of p to each query issued by any $q\in Q$ is always a winning response." Intuitively, this means that for $q\in Q$, the link connecting q to p is not among the t slowest links of q.

In dynamic system, A. Mostefaoui [11] defines new system model, and proposes a new notion α to define the number of stable processes, where α is similar to *n-f* in static system. It guarantees that a process can receive at least α responses after it has issued a query. It modifies a static election algorithm [15] to work in this new model. M. Larrea [5] specifies what conditions should be satisfied to elect an eventual leader in a dynamic system. It defines the stability and synchrony condition, points that if no more processes join in or no more processes leave out, an leader should be elected. It proposes an election algorithm based on entering time stamp comparing. To elect an eventual leader in a system with bounded concurrency, it [4] firstly builds a failure detector. However, this failure detector cannot guarantee that all good processes are sorted in the same sequence in different processes. So it extends to add a message exchange procedure to make sure that eventually all good processes are sorted in same order in failure detector modular.

3 System Model and Assumptions

3.1 Processes and Clusters

We consider a dynamic distributed system with a changing set of processes $\Pi=\{p_1, p_2, ...\}$. Processes can join and leave the system freely. A process may also fail by crash. A process that may join the system is unknown to the system before it joins, and it becomes known to the system immediately after its joining.

There are a number *n* of clusters, denoted by $C=\{C_1, C_2, ..., C_n\}$. Each process must be in some cluster during the time it is in the system. A process may change its cluster from time to time due to system dynamicity. Each process knows the set *C* and knows which cluster it is currently in, but it does not know what other processes are in the same cluster.

Please notice that, we assume the existence of clusters but how to define and construct such cluster based hierarchy is out of the scope of this paper. One possible method is to define clusters based on geographical grids or cells as in cellular phone systems. That is, the processes are distributed in a geographical area, clusters can be defined according to physical coordinator or coverage of signals.

Since processes may switch among clusters, the number of processes in one cluster may change from time to time. There is no bound (either upper or lower bound) on the number of processes in one cluster.

3.2 Asynchrony and Stability

The system is asynchronous and there is no time bound on the operation execution time and message delay. The processes can communicate with each other by message-passing. The communication channel is reliable and no message will be lost due to channel failures. A process can receive only messages sent after the join of the process.

As discussed in existing works [11][14], if the system is completely asynchronous and dynamic, no valid operation and computing can be done. To make hierarchical election possible, we need the following notions and assumptions.

Faulty process: a process that may crash.

Good process: a process that never crashes.

Stable process: a good process that never leaves a cluster after it joins the cluster.

Unstable process: a process that is not stable.

Please notice that, a good process is in fact a process that will not crash during the execution of the protocol. A stable process may change its cluster from time to time as unstable processes, but it will eventually keep stay in some cluster until the termination of the leader protocol execution.

Obviously, the number of processes in a cluster may change from time to time, and there is neither upper nor lower bound. Similarly, the number of stable processes also changes as time goes on. Especially, we denote the set of stable processes in a cluster C_i at some time t as $STABLE_{Ci}(t)$. To ensure the possibility of an eventual leader, we assume that the system (not each cluster) starts with at least one stable process .

To cope with the dynamicity of clusters, we define the following assumption about the cluster hierarchy:

Eventual Cluster Stability: eventually (after some time *CST* and *CST* is unknown), at most f of all the n clusters are empty, i.e. no processes in such clusters; and at most s of the clusters are not empty but with less than α members($s<n-f$).

Accordingly, a cluster that eventually has α or more members is called a *stable cluster*, and the set of stable clusters in the system is denoted by *STABLE*. We require that, after some time, a stable cluster have no more process joining.

Another assumption is about the communication asynchrony. As proved in [14], if the system is fully asynchronous, eventual leader is impossible. To cope with such impossibility, we assume the partial asynchrony within a cluster:

Eventual Cluster Synchrony: after some time *EST*, all the message among processes among the same cluster are delivered within a bound of d time units.

Please notice that, although we assume the existence of *EST* and d, their values are not known by processes. This assumption is similar to the definition of partial synchrony proposed in [25].

3.3 The Query-Response Primitive and Associated Assumptions

Following the existing works on leader election in dynamic systems [11], we adopt the query-response communication primitive suited to a dynamic system, which is usually expressed by the following sequence of basic statements:

- broadcast a query message to all the processes;
- wait until responses have been received from a specific number α of processes.

With the cluster-based hierarchy in this paper, query-response is issued in two different layers.

In the lower layer, a process queries processes within the same cluster. In order to ensure that a process issues a query-response is not blocked forever, we require it to wait for only α responses. This is consistent with the α in the definition of Eventual Cluster Stability.

$winning_i(t)$: the set of processes from which p_i has received a winning response to its last query terminated before or at time t. A winning response is the response that is among the α responses p_i accepts.

Then, following [11], we have the assumption on query-response pattern:

$MP_{cluster}$: In a stable cluster Cl, there are a time QST, a stable process p_l, and a set Q of processes (QST, p_l and Q are not known in advance) such that, $\forall t \geq QST$, we have

(1) $Q \subseteq STABLE_{Cl}(t)$

(2) $l \in (\bigcap_{j \in Q} winning_j(t))$

(3) $\forall x \in STABLE_{Cl}(t) : Q \bigcap winning_x(t) \neq null$

In the upper layer, query-response is conducted among clusters (in fact by cluster-head processes elected by the lower layer election). With respect to the cluster hierarchy, we define the query-response primitive in the upper layer as below:

- broadcast a query message to all the cluster-heads (in fact to all the processes because cluster-heads do not know each other);
- wait until responses from (n-f) clusters (i.e. from cluster-heads of (n-f) clusters).

Since the number of clusters is static and known, based on the definition of eventual cluster stability and the assumption on query-response in [15], we have:

MP_{system}: There are a time SST, a stable cluster C_i and a set G of clusters (t, C_i and G are not know in advance) such that, $\forall t \geq SST$, we have

(1) $|G| = f + 1$ and

(2) $C_i \in (\bigcap_{D \in G} winning_D(t))$.

Here, the notation *$winning_D(t)$* is the same as *$winning_i(t)$*, except that the former is defined in terms of clusters.

4 The Hierarchical Eventual Leader Algorithm

4.1 Algorithm Overview and Key Issues Addressed

The algorithm has two layer, corresponding to the cluster hierarchy, and each layer itself is in fact an eventual leader election algorithm. The lower layer is used to elect cluster-head for each cluster, while the upper layer is used to elect the global eventual leader from cluster-heads. Both the two layer algorithms adopt the query-response primitive.

In the lower layer, each cluster itself can be viewed as a dynamic system, and a dynamic leader election is conducted. However, a cluster is in fact only a part of the whole system, the assumptions and models to define dynamicity of the whole system do not hold for a single cluster, and consequently existing dynamic eventual leader algorithms cannot be applied. This is why we define new assumptions for a cluster in Section III. Even so, there is no guarantee that each cluster can correctly elects a stable and unique cluster-head. Additional mechanism is definitely necessary to handle such situations (in Task 4).

```
Algorithm 1-1: The lower layer election

      / /Code for each process $i$
      Init:
        $rec\_from_i = \Pi$;  $log\_date_i = 0$;  $trust_i = \Pi$;
        $leader_i = i$;  $CH = i$;
        set Timer = timeout $=\Delta$;
        $rec_i = \Pi$;  $seqnum_i = 0$;  $accept_i = \Pi$;

      ////////////Task 1: Query-response within a cluster///////////////
101)    while (true)
102)      broadcast $QUERY(i, leader_i)$ to the whole cluster;
103)      wait until $RESPONSE(rec\_from)$ received from $\alpha$ processes;
104)      $rec\_from_i$ =the set of the senders of $RESPONSE$ at line 103；
105)      $RECFROM_i$ = the union of all the $rec\_from$ in $RESPONSE$；
106)      $trust_i = trust_i \cap RECFROM_i$
107)      if $trust_i$ is modified at line 106 then
108)          broadcast $TRUST(trust_i, log\_date_i)$ to the whole cluster;
          ************update the cluster-head*********
109)      if $trust_i == null$ or $trust_i == \Pi$ then $CH=i$;
110)      else $CH=\min(trust_i)$;
        endwhile

      ////////////Task 2: Dealing with QUERY message //////////////
201)    upon $QUERY(j, leader_j)$ is received from $j$
202)      send $RESPONSE(rec\_from_i)$ to $j$
203)      if $CH==j$ then $leader_i=leader_j$;

      ////////////Task 3:Dealing with TRUST message/////////////////
301)    upon reception of $TRUST(trust_j, log\_date_j)$ from $j$
302)      if $log\_date_j == log\_date_i$ then
303)         $trust_i = trust_i \cap trust_j$;
304)      if $log\_date_j > log\_date_i$ then
305)         $trust_i = trust_j$; $log\_date_i = log\_date_j$;
306)      if $trust_i = null$ then
307)         $trust_i = \Pi$;  $log\_date_i = log\_date_i + 1$;
```

In the upper layer, the number of clusters is known and unchanged, so it is similar to a static system with a fixed set of processes, if each cluster is viewed as a process in the election (by letting only the cluster-heads participate in upper layer election). However, before a cluster become stable and generate one unique cluster-head, more than one cluster-head may exist in the same cluster. This makes existing algorithm for static system not workable. By addressing such difficulty, we design our new algorithm.

The operations of our algorithm are divided into six tasks. The first four tasks compose the lower layer algorithm, while the upper layer algorithm consists of the other two tasks.

4.2 The Lower Layer Algorithm

The lower layer algorithm consists of four tasks. Task 1 to Task 3 are used to do the basic query-response operations, while Task 4 is specially designed to handle the blocking upon response messages in clusters that are not eventually stable. Algorithm 1-1 shows the pseudo code of Task 1, Task 2 and Task 3, and Algorithm 1-2 shows the pseudo code of Task 4.

To execute the lower layer protocol, each process needs to maintain the following data structures.

rec_from_i : The set of processes from which p_i receives a *RESPONSE* message.
$trust_i$: candidate cluster-head set.
log_date: a logical time defines the age of $trust_i$.
$leader_i$: the current global leader process.
CH_i : The local cluster-head of p_i.
timeout: The timeout value of a Timer to detect whether p_i is blocked.
$RECFROM_i$: The union set of *rec_from.*

Two different Types of messages are used to exchange information in the lower layer.

QUERY(i, $leader_i$): A message to query cluster-members
RESPONSE(*rec_from*): The response message to *QUERY.*

Task 1 is the basic query-response part for processes to exchange their information and update their cluster-head candidates. At the beginning, a process p_i broadcasts a *QUERY* message, which contains its id and its global leader's id, to all the members in the same cluster. Then p_i waiting for response messages from other cluster members.

The *QUERY* message is handled by Task 2. When some process j receives the *QUERY* message from i, it will send back a *RESPONSE* message to share its *rec_from* list . Furthermore, if i is j's cluster-head, j will update its global leader by following i. That is, a cluster member will accept the leader elected by its cluster-head.

Again in Task 1, after i receives *RESPONSE*(*rec_from*) from α processes, it will stop waiting. The data structure rec_from_i contains the ids of such α processes. On the other hand, p_i unites all the *rec_from* carried by the RESPONSE messages into a set. If a process is in *RECFROM* , it means that it is still be trusted by at least one process. Then, p_i updates its $trust_i$ against *RECFROM* to get the processes which are trusted by all the processes in the cluster. If $trust_i$ is changed, p_i broadcasts *TRUST*($trust_i$,log_date_i) to the whole cluster.

Task 3 handles the *TRUST* message. When a process j receives *TRUST*($trust_i$, log_date_i) from i, it will compare log_date_j with log_date_i, which are logical ages of the trust sets at p_j and p_i respectively. If the two set are the same old, p_j will update its own trust set to the intersection. If the trust set of p_i is newer, p_j will accept $trust_i$

directly. After this, if $trust_j$ becomes empty, p_j resets its trust set to the whole process set and increase log_date_j by one.

Algorithm 1-2: Handling unstable clusters

```
     / /Code for each process i
     Task 4:Dealing with process blocked
         upon timeout for line 103 occurs
401)         if not enough RESPONSE messages received then
402)             timeout=timeout+1;
403)             CH=i;
404)             set Timer = timeout and goto line 102
405)         else if CH changes at line 109 or 110 then
406)             timeout=timeout+1;
407)         set Timer = timeout;
408)
```

Task 4 is a special task to handle clusters that are not eventually stable, i.e. clusters with less than α member processes, even after the system becomes stable. In such a cluster, a process cannot receive α *RESPONSE* messages at line 103, and it will keep to be blocked. Then, the process cannot execute election successfully, and cannot getting leader information from upper layer processes.

To address such a problem, we add the following operations. A timer is set for line 103. When timeout occurs, the process will check whether it has received *RESPONSE* messages from α processes. If it has not got enough *RESPONSE*, it will increase its timeout value by one, and elect itself as the cluster-head. It will then start a new round a query-response and at the same time join the election in upper layer. On the other hand, if p_i has got enough *RESPONSE* when timeout occurs but its cluster-head is changed at line 109 or line 110, it still needs to increase the timeout value by one.

With the Task 4 above, the processes in clusters with not enough processes will eventually become cluster-heads and they will participate in the upper layer election. Then, these processes will eventually elect the global leader.

Although Task 4 is designed for not stable clusters to avoid blocking at line 102, all clusters (i.e. processes in all clusters) need to execute Task 4, because no one can detect whether a cluster is stable or not. Fortunately, executing Task 4 in stable clusters will not affect the normal execution. This is achieved by the eventual cluster synchrony property. Please refer to the proof section for details.

4.3 The Upper Layer Algorithm

The upper layer algorithm has two tasks, i.e. Task 5 and Task 6, as shown in Algorithm 1-3. The steps are similar to, but not the same as those in the lower layer algorithm because both layer do election using query-response primitive. However, they two layer algorithms are not the same.

Algorithm 1-3: The upper layer election

```
     / /Code for each process $i$
     /////////////Task 5:The query and response of upper layer//////
501)  while ($true$)
502)    if $CH==i$ then
503)      broadcast $ALIVE(C_i, i, accept_i, seqnum_i)$;
504)      wait until $RES(C_k, k, rec_k)$ from (n-f) clusters;
505)      $rec_i$= the set of $C_k$ from which $p_i$ received a $RES$ at line;
506)      let $CLUSTER = \cup$ of all the $rec_k$;
507)      $accept_i = accept_i \cap CLUSTER$ ;
508)      if cluster $j$ has more than one cluster-heads (by checking the
509)      $RES$ received at line 504 then
510)        $accept_i = accept_i - \{j\}$;
511)      if $accept_i == null$ then
512)        $leader_i = i$;
513)      else
514)        $leader_i$ =the cluster-head of min($accept_i$);
     endwhile

     ///////////Task 6:The upper layer message dealing////////
601)   upon reception of $ALIVE(C_j, j, accept_j, seqnum_j)$ from $j$
602)     if $seqnum_j == seqnum_i$ then
603)       $accept_i = accept_i \cap accept_j$;
604)     if $seqnum_j > seqnum_i$ then
605)       $accept_i = accept_j$; $seqnum_i = seqnum_j$;
606)     if $trust_i = null$ then
607)       $accept_i = \Pi$;   $seqnum_i = seqnum_i + 1$
608)     if $C_j == C_i$ && $j != i$ then
609)       $accept_i = accept_i - \{C_i\}$;
610)     else
611)       send $RES(C_i, i, rec_i)$ to $j$
```

In the upper layer algorithm, the following data structures are used:

$accept_i$: the set of candidate leaders.

$seqnum_i$: a logical time defines the age of $accept_i$.

rec_i: the set of (C_k, k) from which p_i received a *RES* message.

CLUSTER: the union set of C_k.

Also, two messages are used to exchange information in the upper layer.

$ALIVE(C_i, i, accept_i, seqnum_i)$: a message to gossip with other cluster-heads.

$RES(C_k, k, rec_k)$: the response message of *ALIVE*.

If a process i finds itself to be a cluster-head, it will participate in upper election by executing Task 4. It firstly broadcasts an *ALIVE* message . For the other hand, as Task 5 describes, if process j receives an *ALIVE* message, firstly it will do the same operation as Task3 to modify *accept* and *seqnum* values. And then it will check whether this message is from some other cluster-head of the same cluster. If they are from the same cluster, process j will erase its cluster id from *accept*. Otherwise, it will send back a *RES* message.

After receiving *RES* messages from *n-f* clusters, process *i* records the set of cluster id it has received a *RES* message so that null cluster will not be considered. Then it calculates the union of these *n-f* rec_k and intersects it with *accept*. It also erases those clusters with more than one cluster-heads from the set of *accept*. Finally ,it will elect its leader according to *accept*.

Compared with Algorithm 1-1, there are two major differences in Algorithm 1-3. In Task 1, a process waits to receive messages from enough processes. In Task 5, because a cluster may have more than one cluster-heads, we use cluster id to represent a process. That is, a process waits to receive messages from enough clusters rather than enough processes. If a process receive messages from different cluster-heads which are from a same cluster, we simply accept the first message received.

Another difference is that, the query message *ALIVE* contains more information than *QUERY* in Task 1. The value *accept* is similar to *trust* and *seqnum* is similar to *log_date* of Task 3. Then, in Task 6, when a process receives an *ALIVE* message, it will check and update its *accept* and *seqnum* information, which are not necessary in Task 2.

5 Correctness Proof

In this section, we prove that the algorithm described in the previous section is correct, i.e. it can eventually elect a unique and stable process as the leader.

In the following proof, we will use the assumptions defined in Section III. In the assumptions of eventual cluster stability, eventual cluster synchrony, $MP_{cluster}$ and MP_{system}, there is a time moment after which the stability or synchrony holds. Although the stabilization time is different for these assumptions, there is a time *GST*, after which all the assumptions hold. We call *GST* the global stabilization time.

Lemma 1. *For each stable cluster C_l, there is a time t and a stable process pl in the cluster such that, after t, very RECFROM contains p_l.*

Proof: With $MP_{cluster}$, after *GST*, p_l is a winning process for all processes in *Q* and each stable process in C_l will receive a winning *RESPONSE* from some process in *Q*. Then, p_l will be included in one of winning *RESPONSE* at each stable process in C_l, the lemma holds. □

Now, we define the relationship "association", and give a claim based on association, which will be used in later proof. The proof of the claim is omitted.

Association: if a process p_i has log_date_i=*log_date* and $trust_i$=*trust*, we call *trust* is associated with *log_date*. Please notice that a *log_date* may be associated with multiple processes' *trust.*

Claim 1. *Let us assume that null is associated with log_date_{max}. Then: (1) a process p_i executes the reset statement at line 307, and $(accept_i, log\_date_{max})=(\Pi, log\_date_{max}+1)$; (2) the pair $(\Pi, log\_date_{max}+1)$ is broadcasted to all processes of the same cluster.*

Lemma 2. *For each stable cluster C_l,* $\exists M, \exists t,$ such that, $\forall t' \geq t, \forall i \in STABLE_{Cl}(t') \Rightarrow trust_i = M$.

Proof: Let t_1 be the time mentioned in Lemma 1, after which no more processes will join in C_l, t_2 be the time that all the faulty processes have crashed and all the messages sent by faulty processes have been delivered.

Let $t = \max(t_1, t_2)$, and log_date_{max} be the max *log_date* value of the processes in $STABLE_{Cl}(t)$. We then prove $M = log\_date_{max}$ or $M = log\_date_{max}+1$.
Suppose *trust* is associated with log_date_{max}. Let p_i has $log\_date_i = log\_date_{max}$ and $trust_i = trust$. There are two different cases.

Case 1): All $trust_i$ is never equal to *null*. Then, no process executes the reset statement, and $trust_i$ cannot be *null* and log_date_{max} cannot be increased. With the communication in Task 3, all stable processes of C_l will have $log\_date = log\_date_{max}$. Obviously, $M = log\_date_{max}$.

Case 2): *Some* $trust_i$ is set to *null*. Then, p_i resets $trust_i$ to Π increases log_date_i by one, and broadcasts it to all the processes. After receiving such a message, other processes will reset their *trust* to Π and update *log_date* to $log\_date_{max}+1$. By Lemma 1, p_l is contained in *RECFROM* of each stable process, then *trust* will not be *null* any more. So, no *log_date* will be larger than $log\_date_{max}+1$, and $M = log\_date_{max}+1$. □

Theorem 1. *For each stable cluster C_l, by the assumption $MP_{cluster}$, all stable processes in C_l eventually elects the same cluster-head.*

Proof: Let $PL = \bigcap \{trust_i, p_i \in STABLE_{Cl}(t), trust_i$ is associated with $M\}$, where t and M is mentioned in Lemma 2.

By Lemma 2, after time t, *log_date* will not increase, so no *trust* is *null*. Then, *PL* != *null*. By Lemma 2, after time t, all the faulty processes already crash and they cannot be contained in *trust* set any more. Then, $PL \subseteq STABLE_{Cl}(t)$. With the communication and update pattern of *trust* in *Task* 3, for process p_i, $trust_i$ = =*PL*.

Since each stable process in C_l has the same *trust* set, and select $\min(trust_i)$ as its cluster-head, the lemma holds. □

Lemma 3. *There is a time t and a stable cluster C_l such that, after t every CLUSTER contains C_l.*

Proof: Similar to *Lemma* 1, With MP_{system}, after *SST*, it holds.□

Claim 2. *Let us assume that null is associated with $seqnum_{max}$. Then: (1) a process p_i executes the reset statement at line 607, and $(accept_i, seqnum_{max}) = (\Pi, seqnum_{max}+1)$; (2) the pair $(\Pi, seqnum_{max}+1)$ is broadcasted to all cluster-heads.*

Lemma 4. *For each cluster-head p_i of any cluster,* $\exists N, \exists t, \forall t' \geq t, seqnum_i = N$.

Proof: Let t_1 be the time mentioned in *Lemma* 3, t_2 be the time all processes of any unstable cluster become cluster-heads. Let $t = \max(t_1, t_2)$. Then, by Claim 2, the proof is similar to that for Lemma 2. □

Theorem 2. *In any execution that satisfied MP_{system} assumption, the protocol described in Algorithm 1-3 implements a leader facility in a dynamic system.*

Proof: Given a run that satisfies MP_{system}, let $PL=\bigcap\{accept_i\}$, where p_i is a cluster-head of cluster Ci, $accept_i$ is associated with N, t and N is mentioned in Lemma 4. Then, like in the proof of Theorem 1, we have $PL!=null$ and $PL \subseteq STABLE$. The difference from Theorem 1 is that, after time $\max(CST_{Ci})$, where $Ci \notin STABLE$, *accept* will not have any cluster that is not in *STABLE*. Then, $accept_i=PL$, and $\min(accept_i)$ is a stable cluster with only one cluster-head. So, a unique and stable process is elected as the leader. The theorem holds. □

6 Conclusion and Future Works

In this paper, we study the problem of eventual leader in dynamic systems with processes joining and leaving dynamically. Different from existing works, our design adopts a cluster based hierarchy in electing an eventual leader. Hierarchical design is efficient to reduce communication cost and improve scalability. After defining new assumptions and models on cluster dynamicity, we design a two-layer election algorithm, which address several challenging issues caused by the instability of cluster constitution. The algorithm is proved to be correct by rigorous proof.

Since this work should be the first attempt for hierarchical eventual leader election, further study on this topic is obviously necessary. Possible directions include hierarchical election using other message exchange patterns in non-hierarchical eventual leader election, timestamp based election, and so on.

Acknowledgments. This research is partially supported by National Natural Science Foundation of China (No. 61379157, No. 61202002), Guangdong Natural Science Foundation (No. S2012010010670), and the Pearl River Nova Program of Guangzhou (No. 2011J2200088).

References

1. Aguilera, M.K., Delporte-Gallet, C., Fauconnier, H., Toueg, S.: Communication Efficient Leader Election and Consensus with Limited Link Synchrony. In: PODC 2004, pp. 328–337 (2004)
2. Hutle, M., Malkhi, D., Schmid, U., Zhou, L.: Chasing the Weakest System Model for Implementing Ω and Consensus. IEEE Trans' on Dependable and Secure Computing 6(4), 269–281 (2009)
3. Fernández, A., Jiménez, E., Raynal, M.: Eventual Leader Election with Weak Assumptions on Initial Knowledge, Communication Reliability, and Synchrony. In: DSN 2006, pp. 166–175. IEEE Society Press (2006)
4. Tucci-Piergiovanni, S., Baldoni, R.: Eventual Leader Election in Infinite Arrival Message-Passing System Model with Bounded Concurrency. In: EDCC 2010, pp. 127–134 (2010)
5. Larrea, M., Raynal, M., Soraluze, I.: Specifying and Implementing an Eventual Leader Service for Dynamic Systems. Int' J. of Web and Grid Services 8(3), 204–224 (2012)
6. Kuhn, F., Schmid, S., Wattenhofer, R.: A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In: Castrovan, M., Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, pp. 13–23. Springer, Heidelberg (2005)

7. Rowstron, A., Druschel, P.: Storage Management and Caching in Past, A Large-scale, Persistent Peer-to-peer Storage Utility. ACM SIGOPS Operating Systems Review 35(5), 188–201 (2001)
8. Vaze, R., Heath, R.W.: Transmission Capacity of Ad-hoc Networks with Multiple Antennas Using Transmit Stream Adaptation and Interference Cancellation. IEEE Transactions on Information Theory 58(2), 780–792 (2012)
9. O'Dell, R., Wattenhofer, R.: Information Dissemination in Highly Dynamic Graphs. In: Proc. of the 2005 Joint Workshop on Foundations of Mobile Computing, pp. 104–110. ACM (2005)
10. Haeupler, B., Karger, D.: Faster Information Dissemination in Dynamic Networks Via Network Coding. In: PODC 2011, pp. 6–8 (2011)
11. Mostefaoui, A., Raynal, M., Travers, C., et al.: From Static Distributed Systems to Dynamic Systems. In: SRDS 2005, pp. 109–118 (2005)
12. Merritt, M., Taubenfeld, G.: Computing with Infinitely Many Processes. Distributed Computing. In: Herlihy, M. (ed.) DISC 2000. LNCS, vol. 1914, pp. 164–178. Springer, Heidelberg (2000)
13. Chandra, T.D., Toueg, S.: Unreliable Failure Detectors for Reliable Distributed Systems. Journal of the ACM (JACM) 43(2), 225–267 (1996)
14. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of Distributed Consensus with One Faulty Process. Journal of the ACM (JACM) 32(2), 374–382 (1985)
15. Mostefaoui, A., Raynal, M., Travers, C.: Crash-Resilient Time-Free Eventual Leadership. In: SRDS 2004, pp. 208–217. IEEE Computer Society Press (2004)
16. Aguilera, M.K., Delporte-Gallet, C., Fauconnier, H., Toueg, S.: On Implementing Omega with Weak Reliability and Synchrony Assumptions. Distributed Computing 21(4), 285–314 (2008)
17. Gupta, I., Chandra, T.D., Goldszmidt, G.S.: On scalable and efficient distributed failure detectors. In: Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC 2001), pp. 170-179. ACM Press (2001)
18. Yang, Z., Wu, W., Chen, Y., Zhang, J.: Efficient Information Dissemination in Dynamic Networks. In: ICPP 2013, Lyon, France, October 1-4 (2013)
19. Mostéfaoui, A., Mourgaya, E., Raynal, M., Travers, C.: A Time-free Assumption to Implement Eventual Leadership. Parallel Processing letters 16(2), 189–208 (2006)
20. Malkhi, D., Oprea, F., Zhou, L.: Ω Meets Paxos: Leader Election And Stability without Eventual Timely Links. In: Fraigniaud, P. (ed.) DISC 2005. LNCS, vol. 3724, pp. 199–213. Springer, Heidelberg (2005)
21. Guerraoui, R., Hurfin, M., Mostéfaoui, A., Oliveira, R., Raynal, M., Schiper, A.: Consensus in Asynchronous Distributed Systems: A Concise Guided Tour. In: Krakowiak, S., Shrivastava, S. (eds.) Distributed Systems. LNCS, vol. 1752, pp. 33–47. Springer, Heidelberg (2000)
22. Raynal, M.: Eventual Leader Service in Unreliable Asynchronous Systems: Why? How? In: NCA 2007, pp. 11–24 (2007)
23. Wu, W., Cao, J., Yang, J., Raynal, M.: Design and Performance Evaluation of Efficient Consensus Protocols for Mobile Ad Hoc Networks. IEEE Transactions on Computers 56(8), 1055–1070 (2007)
24. Wu, W., Cao, J., Raynal, M.: Eventual Clusterer: a Modular Approach to Designing Hierarchical Consensus Protocols in MANETs. IEEE Transactions on Parallel and Distributed Systems 20(6), 753–765 (2009)
25. Larrea, M., Fernández, A., Arévalo, S.: On the implementation of Unreliable failure detectors in partially synchronous systems. IEEE Transactions on Computers 53(7), 815–828 (2004)

Efficient Resource Provisioning for Mobile Media Traffic Management in a Cloud Computing Environment

Mohammad Mehedi Hassan, Muhammad Al-Qurishi, Biao Song, and Atif Alamri

College of Computer and Information Sciences
Chair of Pervasive and Mobile Computing
King Saud University, Riyadh, Saudi Arabia
{mmhassan,qurishi,bsong,atif}@ksu.edu.sa

Abstract. This paper presents an efficient resource allocation model that dynamically and optimally utilizes virtual machine (VM) resources to satisfy QoS requirements of mobile media traffic in a media cloud environment. It additionally maintains high system utilization by avoiding the over-provisioning of VM resources to services or applications. The proposed VM allocation is mapped into the multidimensional bin-packing problem, which is NP-complete. To solve this problem, we have designed a Mixed Integer Linear Programming (MILP) model, as well as heuristics for quantitatively optimizing the VM allocation. The simulation results show that our scheme outperforms the existing VM allocation schemes in a media cloud environment, in terms of cost reduction, response time reduction and QoS guarantee.

Keywords: Mobile media traffic, cloud computing, resource allocation, linear programming, heuristics.

1 Introduction

In recent years, multimedia cloud computing [1] is becoming a promising technology to provide a flexible stack of computing, storage and software services in a scalable and virtualized manner for media-rich mobile applications [2][3][4]. In this mobile media cloud environment, the virtualization technology is applied to package the required CPU, memory, GPU (graphics processing unit), storage and network bandwidth resources of servers into virtual machines to manage and provision heterogeneous multimedia services and applications at lower cost with minimal efforts. These services include but not limited to image/video retrieval, video transcoding, streaming, video rendering, media analytics, sharing and delivery [5][6][7].

Due to the heterogeneity and mobility of the media services and users, mobile media cloud has brought up the need for an efficient VM resource management to satisfy the QoS requirements of the media services, especially when different atomic media services such as streaming service, video transcoding services, rendering services and so on are composed to meet the customer demands [8]. These services have different QoS requirements, and need dynamic VM resource capacity at run-time [1]. In addition, the processing delay of both the atomic and composite media services at server side under different network conditions makes it difficult to efficiently manage VM resources, while fulfilling QoS demands [1][9][10].

Although there are many researches going on to study various VM resource management techniques in cloud environment [11][12][13][14][15][16][17][18][19], very few of them are suitable for mobile media cloud enthronement (Section 2 provides a detailed survey of these works related to current paper). This is due to the dynamic nature of the multimedia services (atomic and composite) in terms of source dynamicity (time-varying rate-distortion characteristics of the encoded multimedia content), channel dynamicity (time-varying channel conditions), and topology dynamicity (where clients can join or leave in the system at any time). Currently, there exist few researches [20][21][22][23][7] related to VM resource allocation in a multimedia cloud environment. However, most of them [20][23][7] do not take into account the composite mobile media service scenario, which can affect the response time as well as the overall utilization of the physical resources. In addition, they do not consider the multiple VM resource dimensions (i.e. CPU, GPU, memory, storage and network bandwidth) in the resource allocation problem.

In this paper, we tackle the aforementioned challenges of VM resource allocation in a mobile media cloud environment. We propose a VM resource allocation model that optimally allocates VM resources to a set of physical machines/servers by considering the dynamic VM resource requirements for atomic and composite mobile media services. It also ensures the minimum QoS requirements of the mobile multimedia services, while maintaining high system utilization by avoiding over provisioning the VM resources for the services. Several experiments were carried out to validate the efficiency of our proposed VM resource allocation model in mobile media cloud platform. These experiments were conducted for different request patterns of mobile media services in various environments. We have also compared our proposed algorithm with three other existing algorithms in media cloud platform, which comprised of load balancing model [23], queuing model [20], and a round-robin allocation. The results include the performance of cost reduction, response time reduction and a QoS guarantee.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes our VM resource allocation model and heuristics for the current multi-dimensional problem. Section 4 presents experimental results and performance comparisons. Finally, Section 5 concludes the paper.

2 Linear Programming Formulation

The proposed VM resource allocation problem is mapped to the multidimensional bin-packing problem [24], which is NP- complete. In this problem, we have to map several items into the smallest number of bins as possible. Here, each item denotes a tuple, which contains its dimensions. In our scenario, we consider each VM as an item and the dimensions like CPU, memory, storage, network bandwidth and GPU, as its capacities. The target is to find a set of physical machine to host the VMs in an optimized way. The basic concern of a VM allocation is that a physical machine must have enough capacity for hosting the VMs. To reduce the hosting cost, the number of active physical machines needs be minimized. To avoid frequent VM migration, certain amount of CPU capacity needs to be preserved as backup resource for handling workload burst. To reduce the response time, the delay of the composite service needs to be controlled. According to above considerations, we design a linear programming (LP) model for quantitatively optimizing VM allocation into physical servers. The input parameters and variables used in the linear programming formulation are presented in Table 1.

Table 1. Parameters and variables for the VM allocation problem

Parameters	
S	set of physical servers
M	set of virtual machines
R	set of resources (CPU, memory, storage, GPU, network etc.)
$C = \{m_1, m_2, ..., m_n\}$	composite video surveillance service from $m \in R$ VMs
$u_{mr} \in R$	utilization for virtual machine $m \in M$ of resource $r \in R$
$c_{sr} \in R$	capacity for physical server $s \in S$ of resource $r \in R$
$Utlc_{sm}$	resource utilization on physical server
$PerCPU_{sm}$	percentage of CPU utilization on physical server
$delComp_I$	delay of composite video surveillance service I
$\delta_s \in \{0, 1\}$	equals to 1 if physical server $s \in S$ is used, 0 otherwise
$\varphi_{sm} \in \{0, 1\}$	equals to 1 if virtual machine $m \in M$ is allocated to physical server $s \in S$, 0 otherwise

For any composite video surveillance service I that needs to be allocated in the cloud, the LP model is presented in Eq. (1) to (6).

$$\min \sum_{s \in S} \delta_s \tag{1}$$

$$\sum_{s \in S} \varphi_{sm} = 1 \quad \forall m \in M \tag{2}$$

$$\sum_{m \in M} u_{mr}\delta_{sm} \leq c_{sr}\varphi_s \quad \forall s \in S\ , \forall r \in R \tag{3}$$

$$delComp_I \leq T \quad \forall I \subseteq M \tag{4}$$

$$\sum_{s \in S} Utilc_{sm}\delta_{sm} \leq T_1 \quad \forall m \in M \tag{5}$$

$$\sum_{s \in S} PerCPU_{sm}\delta_{sm} \leq T_2 \quad \forall m \in M \tag{6}$$

The objective function in 1 aims at minimizing the number of required physical servers. The constraint in 2 guarantees that each virtual machine is mapped to a single physical server. Equation 3 guarantees that the virtual machine demands allocated in each physical server do not overload its capacity. The constraint in 4 guarantees that the delay of composite video surveillance service I does not exceed a certain threshold value T. Equation 5 helps to improve the overall resource utilization. The constraint in 6 can reduce the chance of CPU overload and can potentially balance the CPU utilization among all physical servers.

The delay of composite video surveillance service $delComp_I$ is defined differently in different scenarios as follows: 1) In the asynchronous composition case, where the VMs have no inter-communication with each other, $delComp_I \leq T$ can be viewed as the combination of delay constrains provided by all VMs. 2) In the synchronous composition case, where the screen update from m VMs is synchronized, $delComp_I \leq T$ means that the most strict delay constraint among all VMs is applied to every VM. 3) In the sequential composition case, where the output of the service running on a predecessor VM is the input of the service running on a successor VM, $delComp_I \leq T$ denotes that the delay constraint on the predecessor VM is incrementally applied to the successor VM.

The resource utilization condition $Utilc_{sm}$ is defined as the standard deviation of the percentage of free CPU, memory, GPU and network bandwidth capacities on physical server $s \in S$ after allocating virtual machine $m \in M$. Applying constraint on resource utilization condition can avoid overuse of any resource. It can greatly improve the optimization results when the workload is highly heterogeneous. The value of T_1 should be periodically updated if the arriving workload varies time to time.

A common constraint on free CPU T_2 is 0.7-0.75, which means the maximum CPU utilization on each physical machine is 70%-75%. The remaining CPU capacity is used to handle unexpected burst. T_2 can be also specified according to the profiling of each application. The applications having frequent burst may require a small T_2 value.

2.1 Heuristics

The proposed VM resource allocation problem can also be solved using heuristics. Although heuristic solutions will not guarantee an optimal solution, the required time to obtain a feasible solution is much shorter than LP. We utilize the best-fit decreasing (BFD) heuristic [24] and modify it to fit into our situation. In this heuristic, a lexicographic order is utilized to sort each VM demand. Following the heuristic definition, the mapping of each VM will then be performed. In the BFD heuristic, the VM will be mapped to the physical server that leaves the least left over space after the mapping between all available physical servers.

Algorithm Description. In this section, we explain the entire procedure of our proposed allocation algorithm.

Step 1: Check whether it is necessary to re-select the threshold value. If it is necessary, then run the threshold selection algorithm; Otherwise, use the previous threshold value.

Step 2: Use the three heuristics FFD, BFD and WFD with the threshold value to generate three allocation schemes.

Step 3: Choose the best allocation scheme and enforce allocation.

Step 4: Update the physical server information for next allocation.

Complexity Analysis. The time complexity of the proposed allocation algorithm is denoted by $O(3 \times n^2m + 9 \times \log_2 100 \times pn^2m)$. n denotes the number of service request, m denotes the number of physical servers, and p denotes the probability that the threshold will need to be re-selected. The time complexity of heuristic allocation algorithm is $O(3 \times n^2m)$, since it calls FFD, BFD and WFD functions where each function takes $O(nm)$ to allocate one service request, and $O(n^2m)$ for the entire allocation. The threshold determination algorithm consists of $\log_2 100$ rounds, and heuristic allocation algorithm is called three times in each round. Thus, each execution of threshold determination algorithm takes $O(9 \times \log_2 100 \times n^2m)$. As it is executed with probability p, the total complexity of threshold determination part is $O(9 \times \log_2 100 \times pn^2m)$

3 Performance Evaluation

In this section, we have presented simulation setup description and conducted several experiments to validate the efficiency of our proposed VM allocation approach as done in. These experiments were conducted for different cases, such as low/high heterogeneity of media tasks, and a large/small media task set. We compared our proposed algorithm with three existing algorithms: a load balancing model [23], a queuing model[20], and a round-robin allocation. The results include the performance of cost reduction, response time reduction and scalability guarantee.

3.1 Simulation Settings

For the simulation, we have created two major simulation components: workload generator and cloud simulator. The workload generator takes responsibility of generating atomic, synchronous and sequential workload for simulation.

The two sub-components are designed to generate individual multimedia service workload and to generate delay constraints for three types of service composition, respectively. The cloud simulator receives multimedia service requests from workload generator and creates VMs with pre-configured CPU capability, memory, GPU and bandwidth. The cloud resources and network environment are simulated by generating physical machines/servers with identical capacity and network latency matrix indicating the network latency between the physical machines/servers, respectively. Finally, seven different resource allocation schemes including proposed method are tested. The results such as the number of active physical machine/server and average delay are collected from PM (physical machine) monitor and network monitor.

Table 2. Details of workload group

Heterogeneity	Number of traces	HI	ACU	AMU	AGU	ANU
Low	50-200	0.17	25.2%	28.36%	30%	30%
High	50-200	0.63	47.28%	48.67%	50%	50%

Table 2 shows simulation parameters, where *HI*, *ACU*, *AMU*, *APU*, *ANU* represents heterogeneity index, average CPU utilization(%), average memory utilization(%), average GPU utilization(%), and the average network bandwidth utilization(%) respectively. *HI*, *ACU* and *AMU* were retrieved from the Technical University of Berlin (TU-Berlin) workload [25], which are normally used by researchers and students to execute computational experiments. To address multimedia service issues in our simulation, we generate the workload by considering several representative multimedia service cases. The first case is Discrete Cosine Transform/Inverse Discrete Cosine Transform (DCT/IDCT), which is very CPU intensive and mostly used in the MPEG and JPEG encoding/decoding. Secondly, image rotation has relatively low demand on CPU capacity, but needs more memory and bandwidth. We also consider video rendering workload with high demand on GPU capability. As the size of image/video can be different between the service instances, the workload should not be generated in an identical way even for the same multimedia service case. Using the above simulation parameters and the patterns of multimedia service, we randomly generate multimedia service requests in each workload. Initially, the capacities of physical servers were assumed to be identical. In this simulation, the number of physical server was fixed to 100.

Table 3 specifies the delay settings, where *IDC*, *SDC*, *IDT*, and *SDT* represents the individual delay constraint on atomic media service, the sequential delay constraint on adjacent media services, the individual delay time on a single server, and the sequential delay time on connected servers respectively. The variation of heterogeneity only affects the delay constraint on media services. While allocating the services, two types of allocation scheme are adopted. In large group allocation case, all media service requirements are assumed to be generated and submitted at one time. On the other hand, small group allocation allows only 1-5 media service requests submission. When the current allocation is done, the following 1-5 media service requests will be created. In both cases, each group of composite media services contains 1-5 services, which can be atomic, synchronous or sequential. The service composition does not introduce more resource consumption, but can change the delay constraint on the services.

Table 3. Details of delay

Heterogeneity	IDC	SDC	IDT	SDT
Low	25ms-35ms	25ms-35ms	5ms-30ms	5ms-15ms
High	15ms-45ms	25ms-45ms	5ms-30ms	5ms-15ms

3.2 Experiments

Several sets of experiments were conducted in the simulation. Firstly, we adopted different request patterns of mobile media services/applications in the experiment (i.e. large/small media service group at Low/high heterogeneous environment) to measure the cost optimization capability. The number of media requests was fixed to 100. Fig. 1 shows the simulation results derived from the large task group at low heterogeneity environment. From the results, we have found that our proposed approach and the load balancing method have the same performance. The optimal allocation sequence is applied by the load balancing method as well as our proposed approach. Since the low heterogeneity requests do not overuse any type of resource, the resource utilization condition threshold we used in our proposed approach does not provide further optimization in this environment. The queuing model performs worse than any other solution as it does not consider virtual machine. Thus, each service request must occupy one

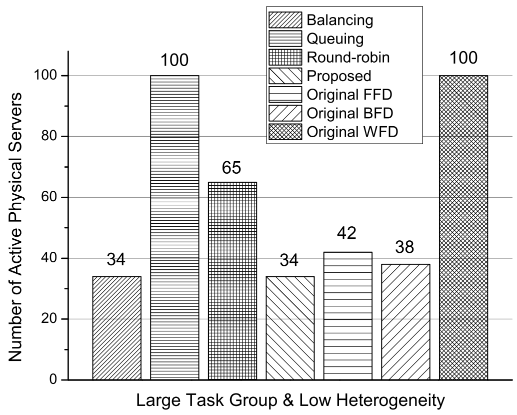


Fig. 1. Cost optimization in large task group at low heterogeneity environment

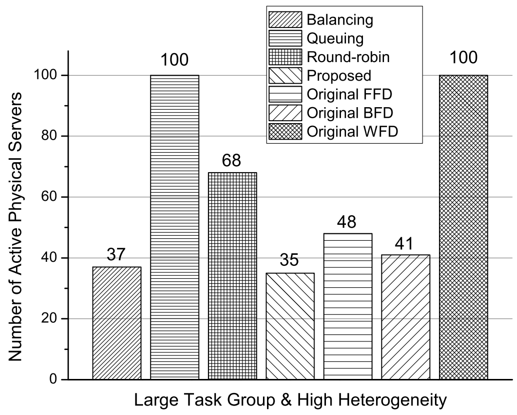


Fig. 2. Cost optimization in large task group at high heterogeneity environment

physical server. The round-robin method randomly chooses physical server for each request. Since it does not provide any optimization method, we have found that 65 physical servers need to be launched according to the allocation results of round-robin method. The original FFD, BFD and WFD are also tested in this simulation environment. The results suggest that those original heuristics considering one dimension (CPU capacity) have inefficient performance while dealing with the cloud resource allocation problem on multimedia service. The original WFD, since it always puts VM on a physical server with highest CPU capacity, actives 100 physical servers to handle 100 service request. The allocation result of original FFD and that of original BFD cause 42 physical servers and 38 physical servers to be active, respectively.

In Fig. 2, we present the results retrieved from the large task group and high heterogeneity environment. Due to the resource utilization condition threshold, our proposed approach outperforms existing algorithms by avoiding overuse of any resource. The performance of load balancing method does not degrade so much, as allocating a large group of requests is easier than allocating several small groups of requests. While allocating a large group of requests the load balancing method can always choose the best allocation among all the possible allocations. As the requests consumes more resources, the results of round-robin, original FFD and original BFD increases to 68, 48 and 41 active servers, respectively. The performance of queuing method and that of orignal WFD remains same as what we present in Fig. 1 due to the reasons we explained before.

In order to validate the scalability of our proposed approach, we varied the number of service requests to test the cost in the small task group and a high heterogeneity environment. Fig. 3 shows the results of the scalability test. When the total number of service requests equaled 50, the performance of our proposed approach was slightly better than that of the load balancing method. By increasing

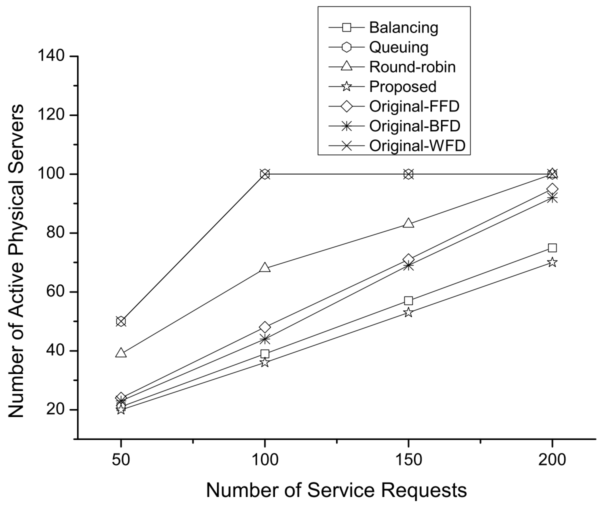


Fig. 3. Scalability Test Results

the workload, a significant difference can be found in the graph. Compared with the load balancing method, our proposed approach demonstrates better scalability. The queuing method and the round-robin method have obvious drawbacks regarding scalability.

In the second set of simulations, we explored the actual response time in different environments. We varied the number of video surveillance service requests from 1-150. In Figs. 4 and 5, the average delay (response time) of the video surveillance services achieved by each solution are illustrated. As can be seen

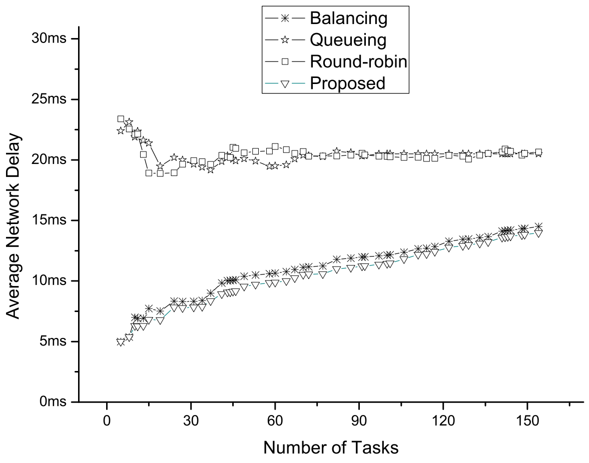


Fig. 4. The response time in low heterogeneity environment

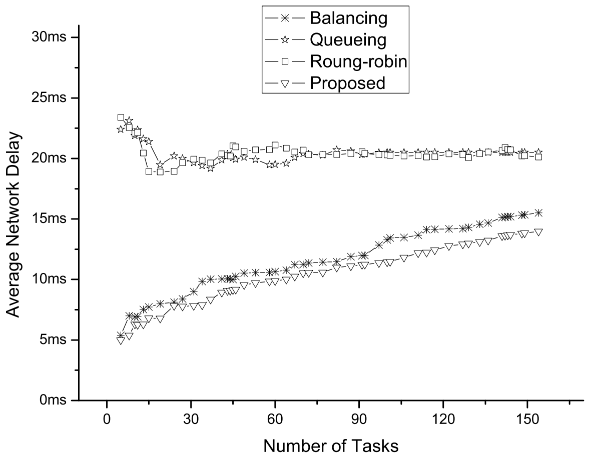


Fig. 5. The response time in high heterogeneity environment

from these figs. the proposed LP and heuristic approaches show their superiority in both low and high heterogeneity environment. The reason why the FracKnap and round-robin performs worse than our proposed approaches is because they do not consider the delay optimization during the VM allocation process. Since we clearly define the delay model for both atomic and composite video surveillance services, we are capable to address the dependency issue among the services and the physical machines. Consequently, we can conclude that our delay model is effective in the composite video surveillance service allocation.

4 Conclusion

The mobile media cloud is emerging as a remarkable technology that can facilitate effective processing of complex multimedia services and provide QoS provisioning for multimedia service or applications from anywhere, anytime and at any device at lower costs. One major challenge for a cloud provider in such mobile media cloud environment is to find an efficient VM resource allocation model for processing media service tasks. This paper presents a VM resource allocation model that dynamically utilizes VM resources to satisfy QoS requirements of media- rich mobile cloud services or applications. In order to do VM resource allocation effectively, we have presented a MILP model, as well as heuristics. Performance comparisons show that our resource management/allocation approach performs very competitively while satisfying users? QoS demand. This work does not include QoE, media play-back quality, media service profiling and benchmarking. As for the future works, we would incorporate some of above as a part of the future work. We believe that our proposed allocation approach can adapt such settings.

Acknowledgment. This work was supported by NSTIP strategic technologies programs, number (12-INF2613-02) in the Kingdom of Saudi Arabia.

References

1. Zhu, W., Luo, C., Wang, J., Li, S.: Multimedia cloud computing. IEEE Signal Processing Magazine 28(3), 59–69 (2011)
2. Amreen, K., Kamal, K.: Mobile cloud computing as a future of mobile multimedia database. International Journal of Computer Science and Communication 2, 29–221 (2011)
3. Simoens, P., De Turck, F., Dhoedt, B., Demeester, P.: Remote display solutions for mobile cloud computing. Computer 44(8), 46–53 (2011)
4. Kumar, K., Lu, Y.H.: Cloud computing for mobile users: Can offloading computation save energy? Computer 43(4), 51–56 (2010)
5. Shi, S., Jeon, W.J., Nahrstedt, K., Campbell, R.H.: Real-time remote rendering of 3d video for mobile devices. In: Proceedings of the 17th ACM International Conference on Multimedia, MM 2009, pp. 391–400 (2009)
6. Dey, S.: Cloud mobile media: Opportunities, challenges, and directions. In: 2012 International Conference on Computing, Networking and Communications (ICNC), January 30-February 2, pp. 929–933 (2012)
7. Li, Y.C., Liao, I.J., Cheng, H.P., Lee, W.T.: A cloud computing framework of free view point real-time monitor system working on mobile devices. In: 2010 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 1–4 (December 2010)
8. Lin, Q., Tretter, D., Liu, J., O'Brien-Strain, E.: Multimedia analysis and composition cloud service. In: Proceedings of the Third International Conference on Internet Multimedia Computing and Service, ICIMCS 2011, pp. 55–58 (2011)
9. Tolia, N., Andersen, D., Satyanarayanan, M.: Quantifying interactive user experience on thin clients. Computer 39(3), 46–52 (2006)
10. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE Pervasive Computing 8(4), 14–23 (2009)
11. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems (2011)
12. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurrency Computat.: Pract. Exper. 24, 1397–1420 (2012)
13. Energy-efficient and multifaceted resource management for profit-driven virtualized data centers. Future Generation Computer Systems (2012)
14. Resource allocation algorithms for virtualized service hosting platforms. Journal of Parallel and Distributed Computing 70(9), 962–974 (2010)
15. Resource management in software as a service using the knapsack problem model. International Journal of Production Economics (2011)
16. Nguyen Van, H., Dang Tran, F., Menaud, J.M.: Autonomic virtual resource management for service hosting platforms. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, CLOUD 2009, pp. 1–8. IEEE Computer Society, Washington, DC (2009)
17. Weiwei, L., Deyu, Q.: Research on resource self-organizing model for cloud computing. In: 2010 International Conference on Internet Technology and Applications, pp. 1–5 (August 2010)

18. Wei, G., Vasilakos Athanasios, V., Yao, Z., Xiong, N.: A game-theoretic method of fair resource allocation for cloud computing services. J. Supercomput. 54, 252–269 (2010)
19. Hassan, M.M., Hossain, M., Sarkar, A., Huh, E.N.: Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform. Information Systems Frontiers, 1–20 (2012)
20. Xiaoming, N., Yifeng, H., Guan, L.: Optimal resource allocation for multimedia cloud based on queuing model. In: 2011 IEEE 13th International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6 (October 2011)
21. Nan, X., He, Y., Guan, L.: Optimal allocation of virtual machines for cloud-based multimedia applications. In: 2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP), pp. 175–180 (September 2012)
22. Xiaoming, N., Yifeng, H., Ling, G.: Optimal resource allocation for multimedia cloud in priority service scheme. In: 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1111–1114 (May 2012)
23. Wen, H., Hai-ying, Z., Chuang, L., Yang, Y.: Effective load balancing for cloud-based multimedia system. In: 2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), vol. 1, pp. 165–168 (August 2011)
24. Kou, L.T., Markowsky, G.: Multidimensional bin packing algorithms. IBM J. Res. Dev. 21, 443–448 (1977)
25. Ferreto, T.C., Netto, M.A.S., Calheiros, R.N., De Rose, C.A.F.: Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. 27, 1027–1034 (2011)

A Community Cloud for a Real-Time Financial Application - Requirements, Architecture and Mechanisms

Marcelo Dutra Ős and Graça Bressan

LARC, University of São Paulo, São Paulo, Brazil
mdutraos@usp.br, gbressan@larc.usp.br

Abstract. Nowadays, Cloud Computing is appealing for both IT businesses, which have a new opportunity to lower costs and expand its computing capacity, as for the scientific community, which has a scalable and elastic computer infrastructure at disposal. The research regarding cloud environments has been a theme of growing interest in the last few years as this new distributed computing paradigm poses challenges for both the application developers and the infrastructure providers. In this paper, it is presented a financial real-time application and a cloud architecture which can address the real-time requirements of this application through the adoption of real-time scheduling techniques. This architecture is based on the concept of a community cloud, where the participants share resources to each other dynamically in a decentralized manner.

Keywords: community cloud computing, real-time distributed scheduling, quality of service.

1 Introduction

Cloud Computing is a distributed computing paradigm which has been extensively applied to many fields of interest in the last few years [6], [18], ranging from ordinary web architectures to very particular scientific applications. The pay-per-use model and ubiquitous access methods have made Cloud Computing an interesting alternative to high-scale and parallel computing as well [12], [18]. As it has been primarily designed to maximize throughput and utilization, some applications, among them real-time ones, poses difficults to cloud environments. In this paper, it is presented a financial real-time application which can take advantage of the benefits offered by cloud environments but which must overcome its real-time limitations. It is proposed a cloud architecture which can address the real-time requirements of this application through the selection of real-time scheduling techniques. Also, it is adopted the concept of community clouds, where one participant can borrow resources to/from another depending on the utilization and the agreements between each one. The rest of this paper is organized as follows: in Sect. 2 it is presented the concepts regarding cloud computing, real-time systems, the relations between them and the limitations and advantages of using cloud environments for real-time applications. In Sect. 3 it is described the real-time financial application - trading securities in stock exchanges - which is the case study for this paper. In Sect. 4 it is presented the concept of community clouds and its applicability to the cloud environment proposed; also, the services envisioned to be offered by this infrastructure are listed. In Sect. 5, mechanisms such as scheduling and QoS (Quality of Service) are proposed to be adopted in this environment. Finally, we conclude with a positioning of this architecture in relation to other works and propose future developments.

2 Concepts

2.1 Cloud Computing

The Cloud Computing paradigm is derived from other computing fields such as Grid Computing [18], Distributed Systems [33], Networks and Operating Systems. Cloud Computing is defined by NIST [20] as "...a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources ... that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud environments present some common properties such as: elasticity - which is the ability to shrink or expand its computational capacity on demand, multi-tenancy - which is the ability to create different environments for distinct customers in the same shared infrastructure, processing of huge amounts of data, loosely-coupled distributed processing, ability to survive hardware failures, virtualization and heterogeneity of both physical and virtual resources and applications [6], [18], [34]. Typically, the service model offered by providers can be classified into the following categories [32], [26]: IaaS - *Infrastructure as a Service*, where low level resources such as CPU, memory, storage and network are offered for consumption; PaaS - *Platform as a Service*, where development platforms are available in the cloud for the creation of applications; and SaaS - *Software as a Service*, where the software itself is offered through the cloud infrastructure.

2.2 Real-Time Systems

A real-time system is constrained by timing deadlines and integrity checking. Also, it should present the following desirable features as described by Giorgio C. Buttazzo in [13]: timeliness - which is the property of executing tasks correctly and within a required deadline, capacity to handle peak loads, predictability, fault-tolerance and modularity. Further, a real-time system can be classified into two types [14]: hard - which presents the property that if one deadline fail a catastrophic condition will occur - and soft - where if one deadline is missed the system will not perform adequately but no damage will affect the surrounding environment. In order to meet deadline constraints, real-time systems employ a wide variety of real-time scheduling algorithms, such as EDF

(Earliest Deadline First) [22]. Examples of real-time applications are as broad as: flight control systems, military systems, robotics, industrial automation and financial applications, among others [14], [28].

2.3 Limitations of Cloud Environments for the Support of Real-Time Applications

Cloud environments were primarily designed to maximize throughput and the utilization of resources. Currently, they present limitations to support real-time applications such as: scheduling algorithms which do not take time into account [31], [29], [19], no support for a real-time clock as an internal time reference [13], delay to provision resources and virtual machines [36], no predictability of the execution of tasks [6] and rudimentary QoS mechanisms [21]. Also, the suboptimal physical topology and connectivity commonly adopted in order to accomodate a large number of applications brings performance penalties in comparison with other distributed systems, as described by R. Aversa in [7].

2.4 Advantages of Adopting Cloud Environments for the Support of Real-Time Applications

Despite these limitations, cloud environments are indeed attractive to the deployment of real-time applications as they present features such as: elasticity, multi-tenancy, ability to survive hardware failures, virtualization support and a layer of abstraction where many different applications could execute its own original code with no knowledge of the underlying platform of software and hardware - which provides flexibility and portability.

3 Real-Time Financial Application

The case study in this paper is a trading application, where the so-called participants - financial institutions, brokerage houses - use to send orders of purchase or sale of securities (most commonly equities, futures and options) to stock exchanges on behalf of their customers. The decision to send orders to buy or sell securities is done on real-time, based on a particular strategy - realized by an algorithm - whose inputs are the prices of the securities at an instant in time. These prices are received by the participants by means of a continuous flow of information called market data [4], which is generated by the stock exchanges and it advertises the absolute and relative prices of the securities being traded. Market data is comprised of two data channels: incremental channel, which provides the delta in prices for the securities since the last trading done and the snapshot channel, which advertises the absolute prices of all the securities. The incremental channel is aperiodic and updates the prices of all securities in real-time. The snapshot channel sends its information in a periodic rate and it adds some delay in comparison with the incremental channel.

At the stock exchange, the orders sent by the participants are stored in a table called the order book, which lists all bids (purchase) and asks (sale) for all the securities. Finally, a matching engine will verify if there is a match between these bids and asks in order to confirm or not a deal. This is illustrated in figure 1.

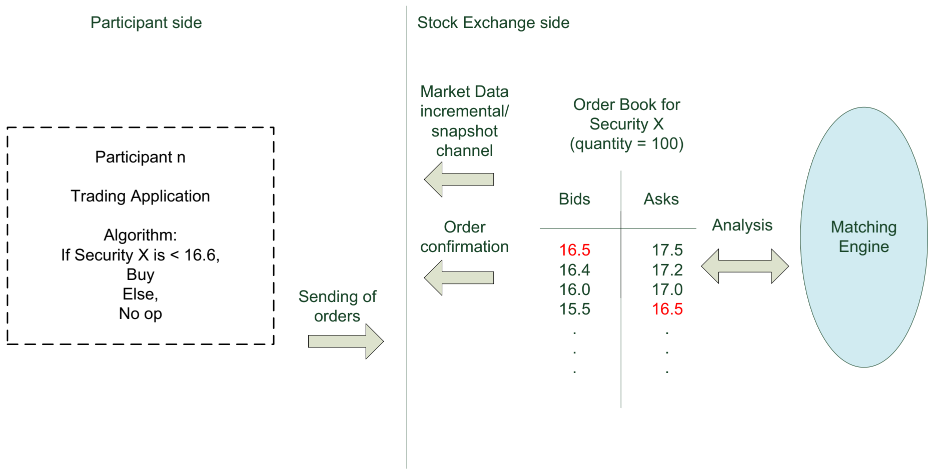


Fig. 1. Trading environment

As the conditions of the market fluctuate, so does the number of orders traded. On busy days, the number of orders can increase to a very high rate. At the stock exchange side, every order should be processed in real-time. The same is true for the participant side, where the decision to generate a new order of buy or sell should be done in real-time.

For instance, consider a particular security X which is traded by a stock exchange, whose price might be increasing during the first hour of a trading day. Conversely, in the following hour, its price might decrease because of a particular market condition. Participant A developed a strategy where it will make money if the price of this security is increasing. On the other hand, participant B developed a strategy where it will make money if the price of this security is decreasing. They do not know about each other's strategy. Some other participants might have the same strategy as participant A or B. The aim of both participants A and B is to be the fastest to send orders to the stock exchanges trading system if the price of the security X is increasing or decreasing, respectively, so they can realize their strategies. Also, they want to do that with the minimal possible investment on infrastructure.

3.1 Typical Parameters of Stock Exchanges

Typically, jobs that need to be run in real-time at the participant side in order to send orders to stock exchanges have the following characteristics: they are aperiodic (event-triggered) as their input is based on market data's incremental channel, they allow parallel execution, they allow preemption for some tasks (depending on its priorization) but not for others, their computation time may vary as well as their deadlines and they might present precedence contraints. Overload conditions (as described in [13]) may occur both because jobs may arrive more frequently than expected or because their computation times might exceed their expected value. Commonly, trading is available during business hours, but there are variations between the opening and closing hours of the market for each product.

The parameters below are of interest for trading:
- Market data's bandwidth: typically in the order of Mbits/s
- Market data's number of messages: the rate can reach thousands of messages per second and millions per day
- Time to process orders at the stock exchange side: typically in the order of microseconds
- Latency from participants to stock exchange venues: typically in the order of microseconds to milliseconds
- Jitter from participants to stock exchange venues: minimum, typically in the order of microseconds
- Number of orders sent by a participant to a stock exchange: typically in the order of thousands per second. Limited by participant
- Numbers of orders that can be processed by a stock exchange system: typically in the order of hundreds of thousands per second and millions a day

As an example, table 1 shows some values measured in three different stock exchanges. BMFBovespa is the stock exchange which is responsible for the trading of equities, futures and options in Brazil. Its statistics can be found at [1] and the numbers in table 1 are related to equities traded on 2014.01.07. BATS Trading is the third largest US equities market (after NYSE and Nasdaq) and its statistics can be found at [9]. The numbers for BATS in table 1 are related to the BZX Exchange on 2012.01.03. CME (Chicago Mercantile Exchange) is the stock exchange in the US responsible for the trading of commodities, futures and options and its statistics can be found at [16]. Its numbers in table 1 are related to the Futures and Options market in June-2012.

Table 1. Typical values for stock exchanges

Parameter / Stock Exchange	BMFBOVESPA	BATS	CME
Market Data - Total Number of Messages	10,779,788	N/A	N/A
Market Data - Peak of messages/s	8,045	361,739	37,000
Market Data - Peak of messages/ms	N/A	6,184	N/A
Market Data - Peak Bandwitdh in Mb/s	5.3576	124	32
Market Data - Peak Bandwitdh in Mb/ms	N/A	0.639	N/A

What can be concluded from these data from different stock exchanges, despite the contrasting order of magnitude that each market deals with is:

1. The number of market data messages has a large variation over time. Standard deviation is high and it depends on the "conditions of the market". So most of the time the hardware allocated by participants are idle. On the other hand, some participants might be prevented from sending orders during overloads because of the lacking of resources' processing
2. The variation of traffic happens as fast as a millisecond (or lower) interval
3. The lowest the latency between the participants and the trading system the better
4. The lowest the jitter between the participants and the trading system the better
5. Customers pay different prices for different physical distances to the trading systems. That's why colocation spaces are the most valuable ones at the stock exchanges' premises. Typically, each customer will deploy its own infrastructure in these colocation spaces, with the price for rental being proportional to the number of rack units allocated. As of today, NYSE (New York Stock Exchange) claims to deliver the lowest latency connection from colocated customers to its trading system - 75 microseconds round-trip time [30]

3.2 Typical Infrastructure Deployed at Colocation Facilities

Typically, a participant will deploy the following infrastructure at a colocation facility: a network connection to the stock exchange which provides low latency, servers with multi-core CPUs, a relatively small storage system which will store all the orders sent and deals done and a number of backdoor connections to other venues which can use mixed information in order to have a global (or partially global) strategy of trading - which is called market arbitration. The participant's system must work in real-time, on the order of microseconds to milliseconds. As each customer deploys its own infrastructure, the stock exchange has to provide enough physical space and power supplies. The same market data information should be received by all customers. Typically, in order to save bandwidth, this is done based on multicast techniques. Figure 2 depicts a typical infrastructure deployed by participants in a colocation rental space at a particular stock exchange's premises. Each participant deploys its own dedicated servers/CPUs, storage and backdoor connections to other venues as well. There is one (sometimes two for the sake of redundancy) network connection to the stock exchange by participant, whereby it will receive market data, send orders of purchase/sale and receive confirmations whether the deals were done.

4 Description of the Cloud Environment Proposed

4.1 Requirements

Given the financial application described, a cloud environment is proposed for participants at colocation spaces in order to provide a real-time infrastructure

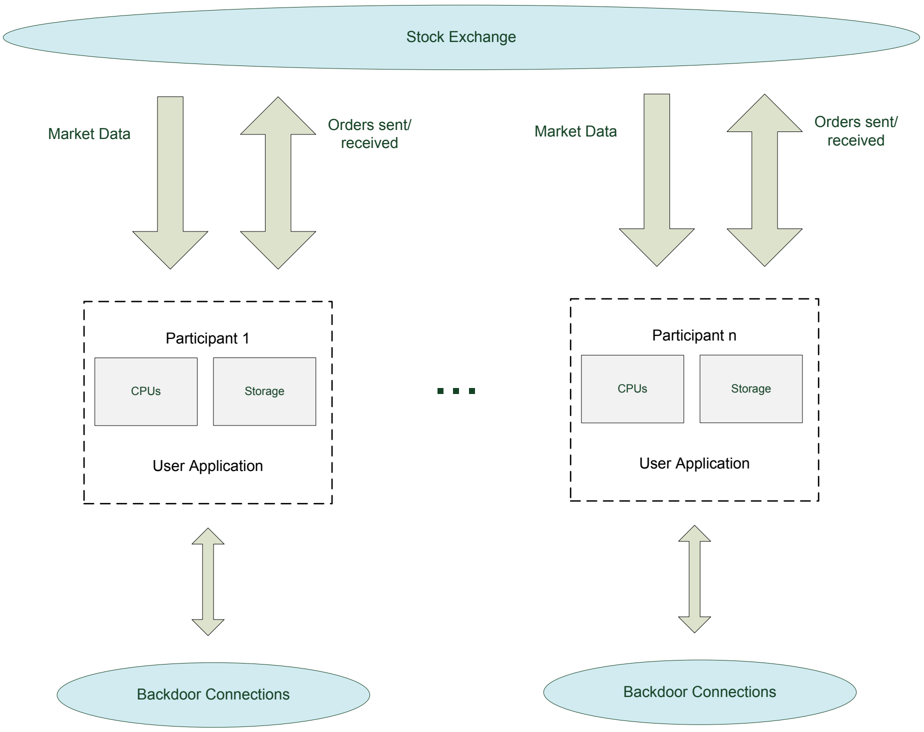


Fig. 2. Typical infrastructure at a stock exchange's colocated rental spaces for trading

to trade securities at stock exchanges. It should meet real-time systems' requirements such as: timeliness, capacity to handle peak loads, predictability, fault tolerance and modularity. Also, it should allow the preemption of tasks.

4.2 Architecture

The cloud environment proposed in this work adopts the concept of community clouds as described by Briscoe and Marino in [10]. Community clouds derive from the concept of digital ecosystems [11] and present properties of adaptation, self-organization, scalability and sustainability, inspired by natural ecosystems, which are robust and scalable too. In this type of architecture, the infrastructure is shared by different companies which present a common goal [32]. In the case of this research, the common goal is trading in real-time. These companies can provide resources to each other in order to increase processing capabilities during peak times, taking on the roles of consumer, producer and (or) coordinator [10]. These resources might be servers which provide CPU cycles, network bandwidth or storage areas which are connected locally to each other in a certain physical topology, which could assume many logical forms. It is envisioned that partner companies might establish community clouds depending on their economic interests; conversely, public ou private clouds do not provide this level of flexiblity and adaptation as they possess a centralized controller or arbitrer who is the owner of the scheduling decisions for the whole system.

The primary goal of such communities is to provide a better performance in terms of orders sent and completed to a stock exchange trading system constrained by a certain deadline for its members in comparison with a standalone deployment. Also, another goal is to have a lower infrastructure cost versus performance ratio, therefore allowing them to save money but still being able to achieve performance goals. Each member in the community cloud must possess a gateway which would be responsible for the negotiation of resources between other members, which might occur dynamically.

We adopt probability values, just as described by Briscoe and Wilde in [11], as the chance that company A might offer resources to company B as demanded. This probability values vary over time. Participants will make agreements with other trusted participants in a dynamic fashion and the probability values chosen will reflect these agreements. So, company A will borrow resources as demanded from company B based on these probability values.

In this way, as depicted in figure 3, participants will form a logical community cloud in order to mutually cooperate and use resources from each other. In figure 3, Pxy is the probability level that X will offer resources to Y. This value is unidirectional meaning that not necessarily Pxy equals Pyx. Also, the sum of Pxi, where i varies from 0 to n-1, equals 1.

It is envisioned that these agreements between participants would follow the same dynamism of the stock exchange market. Cooperation and trust levels could be very high during a time interval but fall to very low values during particular market hours.

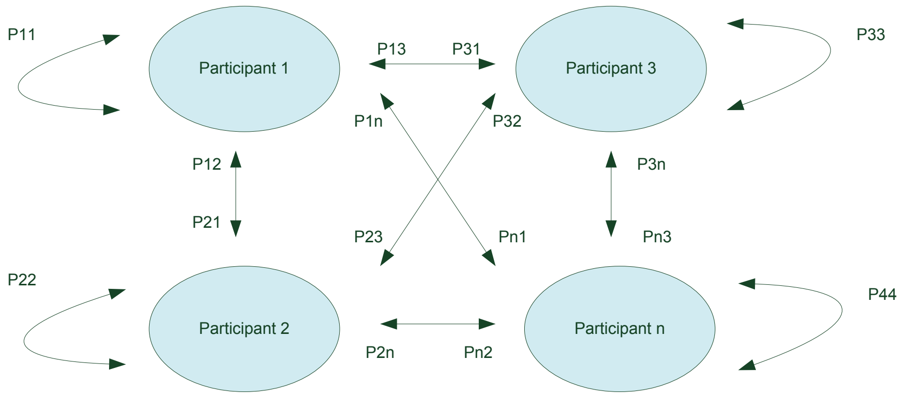


Fig. 3. Community Cloud Environment

4.3 Services

The service provided by this community cloud is infrastructure as a service for real time financial applications - which we call, for short, IaaS-RTF. As an IaaS platform, it should provide: a computer infrastructure which can schedule and execute real-time applications in a timely manner, a network infrastructure which can receive market data traffic and send/receive orders of purchase/sale and a storage infrastructure which can store the deals made and other relevant data. It might be deployed through the use of virtual machines or physical machines that could be allocated depending on the performance expected. Eventually, it could evolve to a Platform-as-a-Service (PaaS) model, where the interested companies would use the same platform in order to develop applications to perform trading.

5 Mechanisms

5.1 Scheduling

In order to match the requirements of real-time applications, clearly a scheduling algorithm which takes into account the deadline of tasks is required. As described in [13], the scheduling problem for the class of tasks described in this financial real-time application is NP-hard, and to find a feasible schedule would be too expensive. To solve this kind of problem, K. Ramamritham and J. Stankovic [23] proposed an algorithm based on a heuristic function H which was adopted in the Spring Kernel [24]. This function H is then calculated to each of the tasks that remain to be scheduled. H can assume many differente values such as: arrival time (as in first come first served), computation time (as in shortest job first), deadline (as in EDF) and many others. An ongoing effort of this research is to propose the most suitable function H for this environment and use the Spring algorithm [23], [24] to solve it. Despite the fact that a heuristic scheduling approach is not optimal, this algorithm can have a complexity close to O(n) (even if most of times reaches $O(n^2)$[13]). Cloud environments present scheduling mechanisms not suitable to enable real-time applications, as their main target is to increase the total throughput of tasks executed during a certain time interval [31], [29], [19].

5.2 Quality of Service

As proposed by John D. Day in [17], the separation of mechanism and policy is a useful paradigm in the modeling of quality of service. In this section, it is listed the QoS mechanisms which are relevant for this environment: admission control - in order to maintain systems' predictability, priorization of tasks - as tasks local to a participant's environment should have more priorization over tasks not belonging to a local environment and discarding of tasks - in order to discard tasks which could decrease the system's overall performance. Last but not least, the system must provide guarantee of execution of tasks under desired parameters as delay and jitter. The tasks to be sent to the system should contain a metadata in order to describe its QoS parameters.

5.3 Adaptive Topology Mechanisms

In HPC environments, the physical network topology is designed taking the application's requirements as the input. So, the physical network does not present performance problems to the applications. Conversely, it provides no flexibility as the hardware is tighly coupled with the software. In a cloud environment, flexibility is the rule. Then, the physical network might become a bottleneck for the deployment of real-time applications. R. Aversa [7] suggests that the main performance issue of cloud environments is not the virtualization overhead, but the poor network connectivity between virtual machines which might bring latency and jitter issues.

In order to avoid this, it should be developed a mechanism where the physical or virtual network should be adapted in order to provide the best connectivity topology for the environment and its applications. It is envisioned that the concept of SDN (Software-Defined Networking) [5] could be used, where a central controller would verify the performance of the applications according to the topology of the network and rearrange it as necessary. Protocols such as OpenFlow [8] could be used to control both the physical and virtual topology improving the applications' performance as the central controller of the SDN demands. Therefore, tasks sent from one participant to another over the network in the community cloud at the stock exchange's colocation space would have more chance to reach the required performance goals.

5.4 Pre-provisioning of Virtual Environments

The delay involved in the provisioning of virtual machines in cloud environments has been presented by S. Gorlatch in [19]. This remains a challenge for environments which require performance, mainly the ones requiring real-time goals. For the real-time financial application described in this paper, the required time to provision virtual machines is not acceptable. So, in this work the pre-provisioning of virtual environments and its virtual machines is considered. Nevertheless, it is not required that the machines provisioned should be only physical or only virtual. This decision should be done on the behalf of each user and the performance required and the cloud environment should be able to support both.

6 Related Works

L. Phan *et. al* [31] analyses the support of current cloud infrastructures for applications which have strong timing guarantees with strict deadlines. The main objectives of this work are to minimize soft real-time variables: (i) miss rate (the fraction of applications that miss their deadlines) and (ii) the maximum tardiness (the maximum elapsed time from deadline to completion time). Therefore, EDF (Earliest Deadline First) schedulers are adapted to run on cloud environments. Their adaptations to EDF include [31]: "data placement, data distribution (skews), online (potentially bursty) arrival of jobs and data and the inherent precedence relationships among MapReduce tasks". Tasks are constrained by the following restrictions: (i) only non-preemptable tasks are allowed, (ii) each task has the same deadline and release time and (iii) there are no failures in the infrastructure.

S. Liu *et. al* [29] introduces a "novel utility accrual scheduling algorithm for real-time cloud computing services". The real-time tasks are scheduled on-line and non-preemptively. A profit time utility function (TUF) and a penalty TUF are applied together in order to not only reward early completions but to penalize abortions or missed deadlines as well. The TUF model was first proposed by E. D. Jensen, C. D. Locke and H. Tokuda [25] and describes the value or utility of a system when a task is completed. The penalty TUF is calculated because a missed deadline in a task means wasted resources as network bandwidth, storage space and processing power. Then, tasks which will have its deadline missed are discarded as soon as possible. It is shown that this proposal outperforms other traditional scheduling approaches such as EDF, the Generic Benefit Scheduling (GBS) [27] and the Profit/Penalty-aware non-preemptive scheduling proposed by Y. Yu *et. al* in [37].

J. Wolf *et. al* [35] developed a flexible scheduling allocation scheme, known as FLEX. It allows the optimization of a variety of standard scheduling metrics, such as response time, deadline misses and tardiness. However, FLEX is not suitable to be used on the scenario described in this paper because it is a clock-based scheduler, where the instants at which the scheduling decisions should run are defined *a priori* [28].

Also, as these works concern improvements in current public cloud infrastructures, none of them consider the adoption of the concept of community clouds.

7 Conclusions and Future Works

7.1 Conclusions

This aim of this research is to make cloud environments suitable for the task of executing real-time applications. Using the financial real-time application described before as the case study for this research, it provides the following contributions: proposal of a decentralized architecture for the participants of this environment based on the community cloud paradigm, development of a heuristic funtion H which is the basis of the scheduling algorithm for this environment, and the researching of mechanisms and policies for the adaptation of the physical and virtual networks in order to improve the performance of the whole environment. Below the future works regarding this research are described.

7.2 Scheduling

The scheduling algorithm based on the heuristic function H being researched considers only the computation of tasks with no precedence constraints. As tasks with precendence constraints may arrive in this environment, a solution for this problem should be studied. In particular, and to make the computation easier and faster, it is being researched the adoption of the method described by H. Chetto, M. Silly and T. Bouchentouf [15], where tasks that have precedence contraints are transformed into tasks that do not. Another advancement would be to consider the use of resource access protocols in order to control the access to shared resources. This work still lacks this kind of mechanisms as it is assumed that no conflict occurs between tasks which attempt to access the same resources.

7.3 QoS

It is being researched which QoS policies should be applied to the QoS mechanisms listed above and under which circumstances. The admission control mechanism can adopt different policies depending on parameters such as hour of the day or occupation of resources. Also, it should be performed a comparison between the effects of having a uniform QoS policy for all participants against a model where each participant adopt its own unique policy.

7.4 Simulations

It is being developed simulation scenarios in order to validate the architecture and the different values the heuristic function H could have as presented in this paper. These simulations have as input some typical parameters of stock exchanges as listed before and it will be evaluated taking into account the following variables:
- Workload variables: level of parallelization of jobs, jobs' typical run-time, jobs' arrival time distribution, jobs' deadline (which should be inversely proportional to market data rates) and priorization between jobs. These variables should be defined in particular for each participant in the community cloud.
- Environment variables: number of participants in the community cloud (minimum equals 2 participants), number of resources provided by each participant, processors' capacity of each participant, different probabilities of borrowing resources between the participants, different logical arrangements between the participants and the average latency incurred in a job migration process.
- Scheduling strategy: it will be simulated different values that the heuristic function H could assume and whether all the participants should adopt the same value or not.
It is still to be defined which environment is more suitable for these simulations: CloudSim [2] or SimGrid with extensions for cloud environments [3].

References

1. http://www.bmfbovespa.com.br/market-data/estatistica.aspx?idioma=pt-br
2. http://www.cloudbus.org/cloudsim
3. http://simgrid.gforge.inria.fr/
4. Market data, http://en.wikipedia.org/wiki/Marketdata

5. Open networking foundation, https://www.opennetworking.org
6. Armbrust, M., et al.: Above the clouds: a berkeley view of cloud computing. Technical report no. ucb/eecs-2009-28, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley (2009)
7. Aversa, R., et al.: Performance prediction for hpc on clouds. In: Cloud Computing, Principles and Paradigms (2011)
8. Balakrishnan, H., et al.: Openflow: Enabling innovation in campus networks. IEEE Network, 6–10 (July-August 2011)
9. BATS Global Markets, Inc.: BATS - US Equity/Options Connectivity Manual, 6.0.2 edn. (March 2012)
10. Briscoe, G., Marino, A.: Digital ecosystems in the clouds: Towards community cloud computing. In: IEEE (corp. ed.) 2009 3rd IEEE International Conference on Digital Ecosystems and Technologies (Dest 2009), pp. 103–108. IEEE (2009)
11. Briscoe, G., Wilde, P.D.: Digital ecosystems: Evolving service-oriented architectures. In: Press, I. (ed.) Conference on Bio Inspired Models of Network, Information and Computing Systems (2006), http://arxiv.org/abs/0712.4102
12. Bryant, R.: Data-intensive scalable computing for scientific applications. IEEE Computing in Science and Engineering, 25–33 (November-December 2011)
13. Buttazzo, G.C.: Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications, 2nd edn. Springer (2005)
14. Cheng, A.M.K.: Real-Time Systems - Scheduling, Analysis and Verification, 1st edn. Wiley-Interscience (August 2002)
15. Chetto, H., Silly, M., Bouchentouf, T.: Dynamic scheduling of real-time tasks under precedence constraints. Journal of Real-Time Systems 2 (1990)
16. CME Group, Inc.: CME Group Client-Managed Router Guidance (July 2011)
17. Day, J.D.: Patterns in Network Architecture - A Return to Fundamentals. Prentice Hall (2008)
18. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proc. Grid Computing Environments, GCE 2008, pp. 1–10 (2008)
19. Gorlatch, S., et al.: Towards bringing real-time online applications on clouds. In: International Conference on Computing, Networking and Communications, Cloud Computing and Networking Symposium, pp. 57–61. IEEE (2012)
20. Grance, T., Mell, P.: The nist definition of cloud computing. In: Special Publication 800-145 (Draft). National Institute of Standards and Technology - U.S. Department of Commerce (January 2011)
21. Perros, H., Xiong, K.: Service performance and analysis in cloud computing. In: 2009 Congress on Services - I, pp. 693–700. IEEE Computer Society (2009)
22. Horn, W.A.: Some simple scheduling algorithms. Naval Research Logistics Quarterly 21(1), 177–185 (1974)
23. Stankovic, J., Ramamritham, K.: The design of the spring kernel. In: Proceedings of the IEEE Real-Time Systems Symposium (December 1987)
24. Stankovic, J., Ramamritham, K.: The spring kernel: A new paradigm for real-time systems. IEEE Software (May 1991)
25. Jensen, E., Locke, C., Tokuda, H.: A time-driven scheduling model for real-time systems. In: IEEE Real-Time Systems Symposium (1985)
26. Lenk, A., et al.: Whats inside the cloud? an architectural map of the cloud landscape. In: ICSE 2009 Workshop, pp. 23–31. IEEE (2009)
27. Li, P.: Utility accrual real-time scheduling: Models and algorithms. Ph.D. thesis, Virginia Polytechnic Institute and State University (2004)

28. Liu, J.W.S.: Real-Time Systems, 1st edn. Prentice Hall (April 2000)
29. Liu, S., et al.: On-line scheduling of real-time services for cloud computing. In: Society, I.C. (ed.) 2010 IEEE 6th World Congress on Services. IEEE (2010)
30. NYSE Euronext: US Liquidity Center Products and Services Guide - Colocation and Networks, 4 edn. (October 2011)
31. Phan, L., et al.: An empirical analysis of scheduling techniques for real-time cloud-based data processing. In: Society, I.C. (ed.) SOCA 2011 Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications, pp. 1–8. IEEE (2011)
32. Rimal, B., Choi, E.: A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing. International Journal of Communication Systems 25, 796–819 (2012)
33. Tanenbaum, A.S., van Steen, M.: Distributed Systems: Principles and Paradigms, 2nd edn. Prentice Hall (2006)
34. Varia, J.: Cloud architectures (July 2008), `http://aws.amazon.com`
35. Wolf, J., Rajan, D., Hildrum, K., Khandekar, R., Kumar, V., Parekh, S., Wu, K.-L., Balmin, A.: Flex: A slot allocation scheduling optimizer for mapreduce workloads. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 1–20. Springer, Heidelberg (2010)
36. Wu, X., et al.: Jump-start cloud: Efficient deployment framework for large-scale cloud applications. Concurrency and Computation: Practice and Experience 24, 2120–2137 (2012)
37. Yu, Y., et al.: Profit and penalty aware (pp-aware) scheduling for tasks with variable task execution time. In: SAC 2010 - Track on Real-Time System, RTS 2010 (2010)

Strategies for Evacuating from an Affected Area with One or Two Groups

Qi Wei[1,2], Yuan Shi[2], Bo Jiang[1], and Lijuan Wang[1,2]

[1] School of Information Science and Technology,
Dalian Maritime University, Linghai Road 1, Dalian, China
qwei2009@hotmail.com
[2] School of Information Science and Technology,
Dalian Institute of Science and Technology, Bingang Road 999-26, Dalian, China

Abstract. This paper considers the problem faced by a group of evacuees who must leave from an affected area as quickly as possible. We seek the strategies that achieve a bounded ratio of evacuation path length without any boundary information to that with. We restrict the affected area to a convex region in the plane, and investigate the problem in two scenarios: general plane and plane in grid network. In these two scenarios, we first present efficient strategies with one or two groups and analyze the competitive ratios of them. In general plane, we give a 30.47-competitive strategy for one group evacuation and a 15.58-competitive strategy for two groups evacuation. In grid network, we give a 33-competitive strategy for one group evacuation and a 19-competitive strategy for two groups evacuation.

Keywords: Computational geometry, Evacuation strategy, Competitive analysis, Convex region, Grid network.

1 Introduction

Motivated by the relations to the well-known searching problem and evacuation problem, much attention has recently been devoted to the problem of how to evacuate from an affected area efficiently when an emergency occurs. In this paper, the affected area is restricted to a convex region in the plane, but boundary information of the area is unknown to affected people. The goal of the evacuees is to reach the boundary of the area as quickly as possible. In some cases, the evacuees may communicate with each other during the evacuation.

In general, the performance of a strategy is measured by a competitive ratio [1], which is defined as follows. Let P denote an affected area. The evacuees are modeled as the points inside P, whose initial positions are all the same. When a strategy *SWI* without boundary information of P is used to evacuate from P, we denote by $|SWI(P)|$ the tour length (cost) of the evacuation groups to evacuate from P by *SWI*. When the evacuees are divided into several groups, $|SWI(P)|$ denotes the longest one.

Let $\left|OPT\left(P\right)\right|$ denote the tour length (cost) required to evacuate from *P* in the case that the evacuees know the boundary information of *P*, i.e., $\left|OPT\left(P\right)\right|$ is the shortest distance between the evacuees and the boundary of *P*. Then, the competitive ratio *r* is defined as follows.

$$r = \sup \frac{\left|SWI\left(P\right)\right|}{\left|OPT\left(P\right)\right|}$$

Previous Work. The evacuation problem has been extensively studied. Chen et al. [2] used the methods of system simulation to compare the evacuation efficiency of three different networks, which include the grid network that we also consider in this paper. Lu et al. [3] and Shekhar et al. [4] studied the shortest path algorithm of evacuation with the consideration of capacity constraints and the increasing number of people in time and space. Berman [5] surveyed the problems of on-line searching and navigation. Burgard [6] considered the evacuate ratio with the cost of strategy as the time spent, instead of path lengths.

The previous studies mainly focus on details of evacuation such as flow and other constraints, so as to analyze the strategy under complete information on the boundary. In a recent work [7], Xu et al. considered a new situation in which the evacuees don't know the boundary information of the affected area. This really occurs in emergency, as the affected region is not known to the evacuees in most case. For the convex region in the plane, Xu et al. [7] gave an evacuation strategy for *k* (≥3) groups of the evacuees with a competitive ratio of $3/cos(\pi/k)$, provided that the evacuees can communicate with each other during the evacuation. For the case *k*=3, Wei et al. improved it to $2+2\sqrt{3}$ [8]. The evacuation strategies for grid network are also studied in [7]. But, their strategies rely on a more parameter *R*, which is the radius of the largest inner circle of the convex region.

Our Work. In the strategy of Xu et al. [7], it is required that the communication among evacuees be always available. In this paper, we further release this restriction, and consider this problem for one group evacuation. This model is more practical. We first present a 30.47 -competitive strategy for one group evacuation in the plane, and a 33-competitive strategy in grid network. Our strategy iteratively follow a half of the boundary of a circle, whose diameter is the length of the distance traveled by that time. The very first diameter is assumed to be on the horizontal line *L* and of length one, which is assumed to be small, as compared to $\left|OPT\left(P\right)\right|$.

Also, we give a 15.58-competitive strategy for two-group evacuation in the plane and a 19-competetive strategy for two-group evacuation in grid network, provided that the evacuees can communicate with each other during the evacuation. Note that the two-group evacuation strategy was not provided in the previous work [7].

The rest of this paper is organized as follows. In Section 2, we give some basic definitions relevant to this paper. In Section 3, we present efficient strategies and analyze the competitive ratio in general plane. In Section 4, we study the evacuation problems for one group and two groups in grid network. In Section 5, we conclude the paper with a discussion of further research and open questions.

2 Preliminaries

In this paper, we define a convex polygon as a closed polygonal chain with all interior angles equal and less than 180 degrees. Also an edge of a polygon is defined as a line segment forming a part of the polygonal chain, a vertex of a polygon as a point where two polygon edges meet and the boundary of polygon as a polygonal chain. Let *P* be a convex polygon and *O* be the origin. The evacuees starting at *O* in *P* don't know the boundary information and their location. The evacuees are divided into several groups to evacuate, and their goal is to leave from *P* as soon as possible. Let $G=\{ G_1,G_{2,\ldots}\ G_n\}$ denote a divided group set of the evacuees.

Successful evacuation of the evacuees requires that all of them have reached the boundary of the affected area. The cost of the strategy is the tour length (cost) $\left|SWI\left(P\right)\right|$ of evacuation group to evacuate from *P* by *SWI* (When the evacuees are divided into several groups, $\left|SWI\left(P\right)\right|$ denote the longest one). The performance of a strategy is measured by a competitive ratio *r* which is defined as above. Our objective is to minimize the competitive ratio *r*.

We now list some properties of the evacuation in this paper:

1. The evacuees starting at *O* in *P* don't know the boundary information and their location, but they can share on-time information all the time.

2. The evacuees move at the unit speed during the evacuation. In grid network, the origin *O* is a node in *P*.

3. When we consider in grid network, assume the network consists of several grid units with edge length of 1. Evacuees travel along the edges of network and cannot stay on the edge but the node of the network.

3 Scenario 1: General Plane

In this section, we study the evacuation problem in the situation of general plane. In reality, the evacuees have several choices for grouping in the evacuation. One is that all the evacuees converge to a group to evacuate from the affected area. The other is that the evacuees are divided into several groups, and choose different routes to escape from the affected area with information sharing among them. The evacuation strategy of *n* groups ($n \geq 3$) had been studied in [7, 8]. Now, we study the strategies of one group and two groups which are more practical in the actual situation.

3.1 Case 1: One Group

In this situation, only one group of the evacuees is required to escape from the affected area. The main idea of our strategy is to iteratively follow a half of the boundary of a circle, whose diameter is the length of the distance travelled by that time. The very first diameter is assumed to be on the horizontal line *L* and of length one, which is assumed to be small, as compared to $\left|OPT\left(P\right)\right|$. An instance is shown in Fig.1, where O denotes the starting position of the evacuees' group, and A_i the point of L intersecting the ith half-circle in our strategy. OA_0 is the first diameter, $|OA_0|=1$. A_0A_1 is the second diameter, $|A_0A_1|= |\overset{\frown}{OA_0}|$. A_1A_2 is the third diameter, $|A_1A_2|= |\overset{\frown}{OA_0}| + |\overset{\frown}{A_0A_1}|$. And so on.

One Group Spiral Evacuation Strategy in General Plane (OSEP)

Step 1: Make a directed line L which pass through O and its direction is arbitrary, as shown in Fig.1. Let $D = \{D_1, D_2\} = \left\{+\overline{L}, -\overline{L}\right\}$ denote the direction set defined by L, where $+L$ is the direction of L, and $-\overline{L}$ is the direction opposite to $+L$. Let j denote the direction and i denote the tour path length that G_1 have walked. At the beginning, let i=0, j=1, k=1 ($i, j, k \in$N) and G_1 starts at point O.

Step 2: Make a segment OA_0 with length 1 on L towards D_1. Make a circle with diameter $d_0=|OA_0|$. G_1 walks along $\overset{\frown}{OA_0}$. $i = i+|\overset{\frown}{OA_0}|$, $j = j+1$.

Step 3: While $j \le 2$, do the followings

1) Make a segment $A_{k-1}A_k$ with length i on L towards D_j. Make a circle with diameter $d_k= |A_{k-1}A_k|$. G_1 walks along $\overset{\frown}{A_{k-1}A_k}$.
2) $j=j+1$, $i=i+|\overset{\frown}{A_{k-1}A_k}|$, $k=k+1$.
3) if $(j > 2)$ let j=1.
4) if (G_1 reach the boundary of P) break.

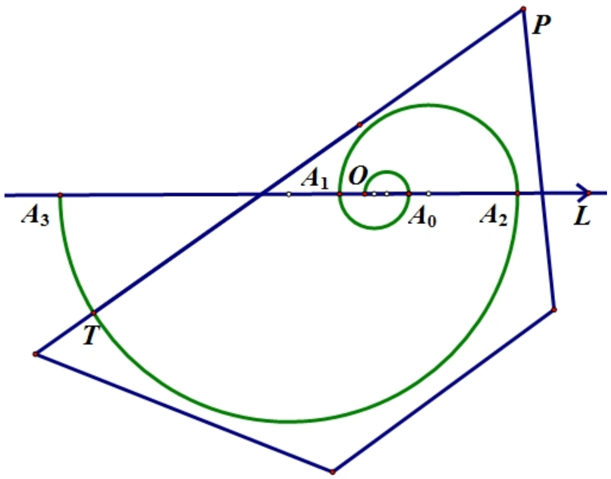


Fig. 1. the strategy OSEP

Theorem 1. In general plane, the evacuate ratio of OSEP is no more than 30.47.

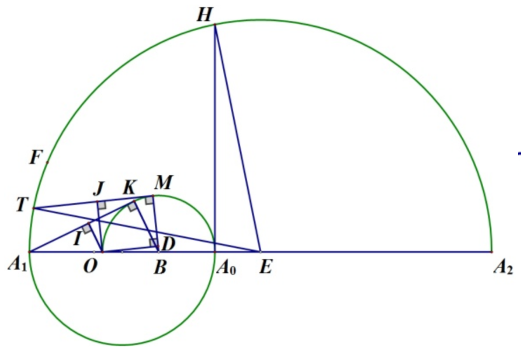
Proof. As described above, the tour path of OSEP consists of several semicircles. Let C={ C_0,C_1,... C_k } and d={ d_0,d_1,... d_k } denote the semicircle set and its diameter set, respectively. Then, $|C_k|$ and $|d_k|$ denote the arc length of C_k and the length of d_k, respectively.

First, consider the relationship between $|d_k|$ and $|d_{k-1}|$ as well as that between $|C_k|$ and $|C_{k-1}|$. With the property of OSEP path, $|d_0|=1$, $|d_1|= |C_0| = \frac{\pi}{2}|d_0|=\frac{\pi}{2}$, $|d_2|= |C_0|+ |C_1| = \frac{\pi}{2}|d_0|+\frac{\pi}{2}|d_1|=|d_1|+\frac{\pi}{2}|d_1|=(1+\frac{\pi}{2})|d_1|=(1+\frac{\pi}{2})*\frac{\pi}{2}$.

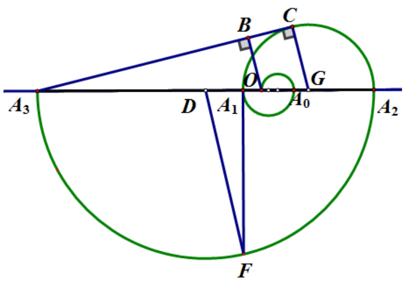
Assume that $|d_k|=(1+\frac{\pi}{2})|d_{k-1}|$ holds when $k\geq 2$, $|d_{k+1}|=|C_0|+|C_1|+\cdots+|C_k|$ $=|d_k|+|C_k|=(1+\frac{\pi}{2})|d_k|$. So, $|d_k|=(1+\frac{\pi}{2})|d_{k-1}|$ holds when $k\geq 2$. Then, $|C_k|=(1+\frac{\pi}{2})|C_{k-1}|$ also holds when $k\geq 2$ because of $|C_k|=\frac{\pi}{2}|d_k|$.

Then, turn to the competitive ratio of OSEP. Let *T* denote the first intersection point between *P* and the tour path. To get the up bound of the ratio, the shortest distance from *O* to the boundary of *P* must be as small as possible. So, the boundary of *P* which has the shortest distance to *O* must approaches but does not meet the tour path at any point except *T*, see Fig.1. For T may be on different locations of the tour path, we distinguish the following situations. (*T* will be not on C_0 or C_1 because of the assumption that the first diameter is smaller than $|OPT(P)|$).

Situation 1. *T* is on C_2. See, Fig.2(a). Let *T* denote any point on the tour path from A_1 to A_2. A_1K, *TM* and HA_0 are tangent lines to C_0 and intersect C_0 at point *K*, *M* and A_0 respectively. Let *F* denote a point on C_2 such that $|OPT(P)|=|OA_1|$ when *T* is at it. Next, we analyze the trend of *r*. If *T* is on $\overset{\frown}{A_1F}$, from A_1 to *F*, $|OPT(P)|$ is a little bit longer than $|OJ|$. The tour path and $|OPT(P)|$ are monotone increasing.



(a)



(b)

Fig. 2. The proof of Theorem 1. (a) Situation 1. *T* is on C_2. (b) Situation 2. *T* is on C_3.

Let x and y denote the radian of $\angle A_1ET$ and $\angle KBM$, respectively. When *T* moves from A_1 to *H*, tangent point *M* moves from *K* to A_0 synchronously. In this process, $0\leq x\leq \angle A_1EH$ and $0\leq y\leq \angle KBA_0$. Because $\angle A_1EH<\angle KBA_0$, $x\leq y$.

$$\angle A_1BK = \arccos\frac{|BK|}{|A_1B|} = \arccos\frac{\frac{|d_0|}{2}}{|d_1| - \frac{|d_0|}{2}} \approx 1.08$$

When $|OJ| \le |BM|$, $|OJ| = |BM| - |BD| = |BM| - |BO| * \cos(\angle A_1BK + y)$. Similarly, when $|OJ| > |BM|$,

$$|OJ| = |BM| + |BO| * \sin(\angle A_1BK + y - \frac{\pi}{2}) = |BM| - |BO| * \cos(\angle A_1BK + y).$$

$$r = \frac{|SWI(P)|}{|OPT(P)|} \le \frac{|C_0| + |C_1| + |\overset{\frown}{A_1T}|}{|OJ|} = \frac{|d_2| + \frac{|d_2|}{2}x}{\frac{|d_0|}{2} - \frac{|d_0|}{2}\cos(\angle A_1BK + y)}$$

$$\le \frac{|d_2| + \frac{|d_2|}{2}y}{\frac{|d_0|}{2} - \frac{|d_0|}{2}\cos(1.08 + y)}$$

$$r_{\max}(y) = r(0) = r_{A_1} = \frac{|SWI(P)|}{|OPT(P)|} \le \frac{|C_0| + |C_1|}{|OI|} = \frac{|d_2|}{|OI|} \approx 14.93$$

So, when T is on $\overset{\frown}{A_1F}$, $r \le 14.93$. (r_{A_1} denote the ratio that T is at A_1 , similarly hereinafter.)

Else if T is on $\overset{\frown}{FA_2}$, from F to A_2, the tour path is monotone increasing but $|OPT(P)|$ is always a little bit longer than $|OA_1|$. So, r is monotone increasing. $r \le r_{A_2} \le \frac{|d_3|}{|OA_1|} \le \frac{|d_3|}{|d_1| - |d_0|} \approx 18.19$. When T is on C_2, $r \le 18.19$.

Situation2. T is on C_3. See, Fig.2(b). FA_1 are tangent lines to C_1 and intersect C_1 at point A_1. A_3C is tangent line to C_2 and intersects C_2 at point C. Next, we analyze the trend of r. If T is on $\overset{\frown}{A_2F}$, from A_2 to F, the tour path is monotone increasing but $|OPT(P)|$ is always a little bit longer than $|OA_1|$. So, r is monotone increasing.

$$r \le r_F < \frac{|C_0| + |C_1| + |C_2| + |\overset{\frown}{A_2F}|}{|OA_1|} = \frac{|d_3| + \frac{|d_3|}{2}\arccos\angle FDA_1}{|OA_1|} \approx 30.47$$

Else if T is on $\overset{\frown}{FA_3}$, from F to A_3, the tour path and $|OPT(P)|$ are monotone increasing. As analyzed in situation 1, we have $r \le r_F$. When T is on C_3, $r \le 30.47$.

Situation3. T is on C_k ($k \ge 4$). When $k \ge 3$, $\frac{|d_k|}{|d_{k-1}|} = (1 + \frac{\pi}{2}) > 2$. So, the centre of C_k and C_{k-1} is on the left and right side of O, respectively. See Fig.1. From A_{k-1} to A_k, the tour path and $|OPT(P)|$ are monotone increasing. As analyzed in situation 1, we have $r \le MAX\{r_{A_{k-1}}, r_{A_k}\}$. When T is at A_3, see Fig.2(b).

$$\frac{|OPT(P)|}{|\frac{d_2}{2}|} = \frac{|OB|}{|GC|} = \frac{|AO|}{|AG|} > \frac{|A_3A_1|}{|AG|} = \frac{|d_3| - |d_2|}{|d_3| - \frac{|d_2|}{2}}$$

When T is at A_{k-1}, we have $\dfrac{|OPT(P)|}{|\frac{d_{k-2}}{2}|} > \dfrac{|d_{k-1}|-|d_{k-2}|}{|d_{k-1}|-\frac{|d_{k-2}|}{2}} = \dfrac{\pi}{1+\pi}$.

$$r_{A_{k-1}} = \frac{|C_0|+|C_1|+...|C_{K-1}|}{|OPT(P)|} < \frac{|d_k|}{\frac{\pi}{1+\pi} * \frac{|d_{k-2}|}{2}} \approx 17.42$$

Similarly, $r_{A_k} < 17.42$. So, when *T* is on C_k (k≥4), $r \le MAX\{ r_{A_{k-1}}, r_{A_k} \} < 17.42$.

All situations had been enumerated above, in term of the intersection point T. The proof is thus complete.

3.2 Case 2: Two Groups

In this situation, the evacuees are divided into two groups, and the communication between the two groups is always available. Both groups follow the same strategy, except that they leave in opposite directions at the first step. Once a group reaches the boundary of *P*, another group goes straight to it. The details of our strategy are described as follows, see Fig.3.

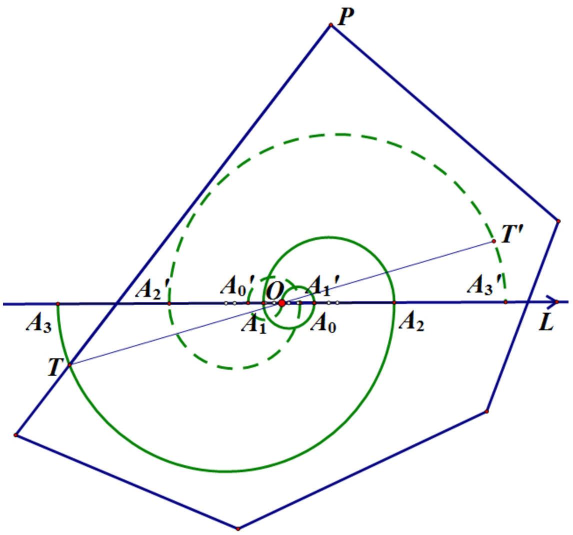


Fig. 3. the strategy TSEP

Two groups spiral evacuation strategy in general plane (TSEP)

Step 1: Make a directed line *L* which pass through *O* and its direction is arbitrary, as shown in Fig.3. Let $D = \{D_1, D_2\} = \{+\overline{L}, -\overline{L}\}$ denote the direction set defined by *L*, where $+\overline{L}$ is the direction of *L*, and $-\overline{L}$ is the direction opposite to $+\overline{L}$. At the beginning, let *i*=0, *j*=1, *k*=1 (*i*, *j*, *k*∈N) and G_1, G_2 start at point *O*.

Step 2: Make two segments OA_0 and OA_0' with length 1 on L towards D_1 and D_2, respectively . Make two circles with diameter $d_0=|OA_0|$ and $d_0=|OA_0'|$, respectively. G_1 walks clockwise along $\overset{\frown}{OA_0}$, and G_2 walks clockwise along $\overset{\frown}{OA_0}'$. $j=j+1$, $i=i+|\overset{\frown}{OA_0}|$.

Step 3: While $j \le 2$, do the followings

1) Make two segments $A_{k-1}A_k$ and $A_{k-1}'A_k'$ with length i on L towards D_j and the direction opposite to D_j, respectively. Make two circles with diameter $d_k=|A_{k-1}A_k|$ and $d_k=|A_{k-1}'A_k'|$, respectively. G_1 walks clockwise along $\overset{\frown}{A_{k-1}A_k}$ and G_2 walks clockwise along $\overset{\frown}{A_{k-1}'A_k'}$.
2) $j=j+1$, $i=i+|A_{k-1}A_k|$, $k=k+1$.
3) if ($j > 2$) let $j=1$
4) if (G_1 or G_2 reach the boundary of P) break.

Step 4: The group that does not arrive at any boundary of P stop and go straight to the intersection point between P and the path of the arrived group.

Theorem 2. In general plane, the evacuate ratio of TSEP is no more than 15.58.

Proof. As described above, the tour path of TSEP consist of several semicircles. Let $C=\{C_0, C_1, \ldots C_k\}$, $C'=\{C_0', C_1', \ldots C_k'\}$ and $d=\{d_0, d_1, \ldots d_k\}$ denote the semi circle set and the diameter set, respectively. As proved in Theorem 1, we also have $|d_k|=(1+\frac{\pi}{2})|d_{k-1}|$ and $|C_k|=(1+\frac{\pi}{2})|C_{k-1}|$ when $k\ge2$. Let T denote the first intersection point between P and the tour path. Let T' denote the position where the other group stands when the first group reaches the boundary of P, See Fig.3. Because the tour path of G_1 and G_2 are symmetric, TT' pass through O and $|OT|=|OT'|$. For T may be on different locations of the tour path, we distinguish the following situations (T will be not on C_0 or C_1 because of the assumption that the first diameter is smaller than $|OPT(P)|$. Suppose that T is on the tour path of G_1, the other case is symmetric.) and also use the analysis method that used in the proof of Theorem 1 to compute the competitive ratio r.

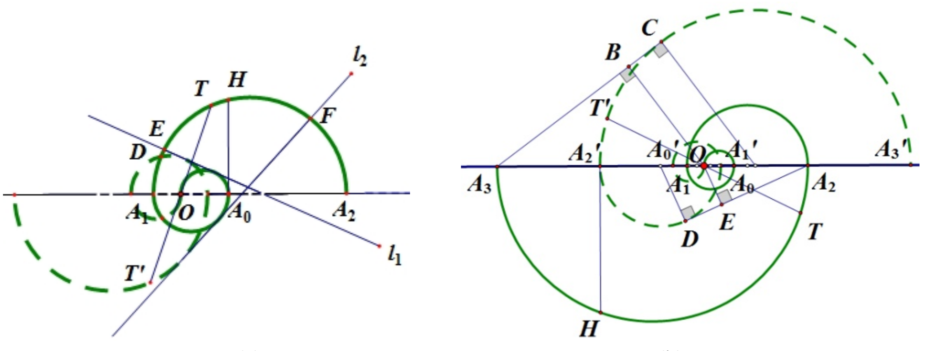


Fig. 4. The proof of theorem 2. (a) Situation 1. T is on C_2. (b) Situation 2. T is on C_3.

Situation 1. T is on C_2. See, Fig.4(a). T cannot be on $\overset{\frown}{A_1D}$, otherwise G_2 will reach the boundary of P before. So, T can be on $\overset{\frown}{DA_2}$. Let l_1 denote the common tangent of C_0 and C_1', and l_2 denote the common tangent of C_1 and C_2'. HA_0 is a tangent line to C_0 and intersect C_0 at point A_0.

If T is on $\overset{\frown}{DE}$, the tour path and $|TT'|$ are monotone increasing but $|OPT(P)|$ is always a little bit longer than the shortest distance from O to l_1. So, r is monotone increasing. $r_{\max}= r_E$. Else if T is on $\overset{\frown}{EH}$, the tour path, $|TT'|$ and $|OPT(P)|$ are monotone increasing.

$$r=\frac{|SWI(P)|}{|OPT(P)|}=\frac{|d_2|+|\overset{\frown}{A_1T}|+|\mathrm{TT'}|}{|OPT(P)|}.$$

As analyzed in the proof of Theorem 1, we have $r_{\max}= r_E$. Else if T is on $\overset{\frown}{HF}$, the tour path and $|TT'|$ are monotone increasing but $|OPT(P)|$ is monotone decreasing. So, r is monotone increasing. $r_{\max}= r_F$. Else if T is on $\overset{\frown}{FA_2}$, the tour path, $|TT'|$ and $|OPT(P)|$ are monotone increasing. $r_{\max}= r_F$.

As discussed above, when T is on C_2, $r_{\max}$=MAX{ r_E, r_F }= r_F = 15.58.

Situation 2. t is on C_k (k≥3). When k≥3, $\frac{|d_k|}{|d_{k-1}|}=(1+\frac{\pi}{2})>2$. So, the centre of C_k and C_{k-1} is on the left and right side of O, see Fig.4(b). From A_{k-1} to A_k, the tour path and $|OPT(P)|$ are monotone increasing. As analyzed in situation 1, we have $r\le r_{A_{k-1}}$ when T is on C_k. $r\le r_{A_{k-1}}\le r_{A_2}<r_F=15.58$.

All situations had been enumerated above, in term of the intersection point T. The proof is thus complete.

4 Scenario 2: Grid Network

Most urban cities have the structure of road network like a grid one [9], See Fig.5. In this section, we study the evacuation problem in the situation of grid network. We establish a rectangular coordinate system coincides with grid network with O as the origin. Let $P(x, y)$ denote coordinates of a point in grid network.

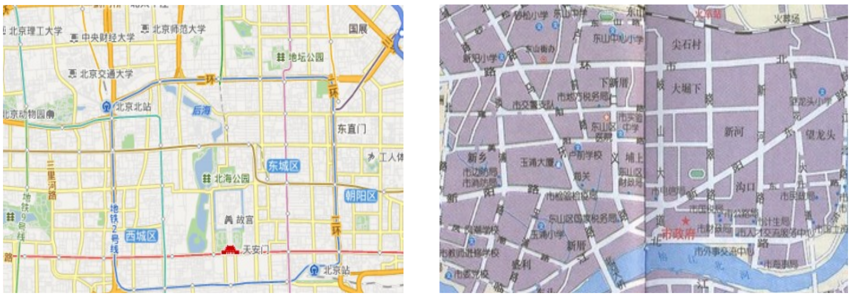


Fig. 5. Part of the traffic network of Beijing and Jieyang, China

Definition 1[7]. For any points $P(x_1, y_1)$ and $P(x_2, y_2)$ in grid network, $L = |x_1 - x_2| + |y_1 - y_2|$ is called evacuation path length.

Let S_{ab} denote the path from a to b on the strategy path, and L_{ab} denote the shortest path from a to b. Without loss of authenticity, several situations may occur in the evacuation. Obviously, in grid network, the strategy EDES[7] is optimal when the group number $n \geq 4$. And what is the optimal strategy when the group number n=1 or n=2? Now, we studied as follows.

4.1 Case 1: One Group

In this situation, all the evacuees leave from the affected area in one group, one path. The main idea of our strategy is to iteratively follow a half of the boundary of a square, whose edge length is the distance travelled by that time. The side length of the first square is two, which is assumed to be small, as compared to $|OPT(P)|$. An instance of our strategy is shown in Fig.6.

One Group Spiral Evacuation Strategy in Grid Network (OSEG)

Step 1: Let $D=\{D_1, D_2, D_3, D_4\}= \{+\overline{y}, +\overline{x}, -\overline{y}, -\overline{x}\}$ denote the direction set in grid network. Let i denote the edge length of the square, j denote the direction, c denote the tour path length that G_1 has walked and k denote the path length that G_1 walks in current move. Let i=2, j=1, c=0, k=0 ($i, j, c, k \in$ N) and G_1 start at point O.

```
Step 2: While ($G_1$ do not reach the boundary of $P$), do the followings
    1) if ($j$==1) $k$=$i$/2; else if ($j$==2) $k$=$i$; else if ($j$==3) $k$= $i$/2; else $k$=$i$.
    2) $G_1$ walks towards $D_j$ with length of $k$. $c$=$c$+$k$.
    3) if (($j$==1&&$i \neq 2$) || $j$==3) {$i$=$c$, $k$=$i$/2, $G_1$ walks towards $D_j$ with length of $k$.
       $c$=$c$+$k$.}
    4) $j$=$j$+1
    5) if ( $j$ > 4) let $j$=1
```

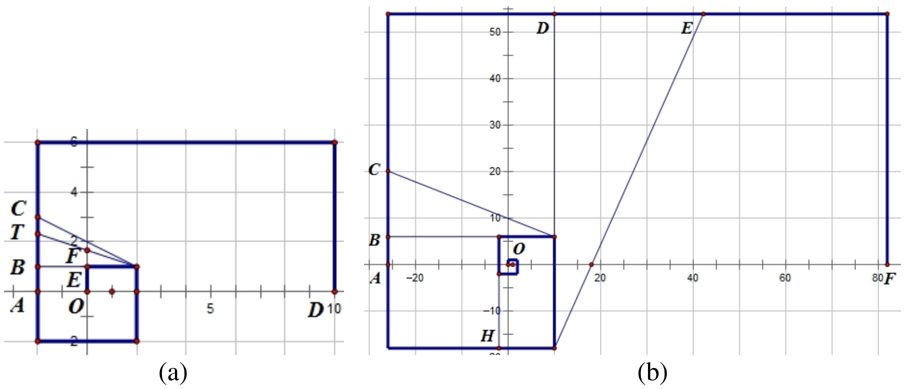


Fig. 6. the strategy OSEG

Theorem 3. In grid network, the evacuate ratio of OSEG is no more than 33.

Proof. As described above, the tour path of OSEG consist of several semi squares, see fig.6. Let $C=\{C_0, C_1, ... C_m\}$ denote the semi square set. Then, $|C_m|$ and $|d_m|$ denote the perimeter of C_m and the edge length of C_m, respectively.

As proved in Theorem 1, we have the relationship between $|d_m|$ and $|d_{m-1}|$ as well as that between $|C_m|$ and $|C_{m-1}|$. $|d_m| = 3|d_{m-1}|$ and $|C_m| = 3|C_{m-1}|$ holds when $k \geq 2$.

Then, turn to the competitive ratio of OSEG. For T may be on different locations of the tour path, we distinguish the following situations. (T will be not on C_0 or C_1 because of the assumption that the first diameter is smaller than $|OPT(P)|$).

Situation 1. T is on C_2. See Fig.6(a). If T is on S_{AB}, the tour path is monotone increasing but $|OPT(P)|$ is always a little bit longer than $|OE|$. So, r is monotone increasing. $r \leq r_B$. Else if T is on S_{BC}, the tour path and $|OPT(P)|$ are monotone increasing. Let x denote $|EF|$, then $|TB|=2x$.

$$r = \frac{|SWI(P)|}{|OPT(P)|} = \frac{|S_{OT}|}{|OF|} = \frac{|C_0|+|C_1|+|AB|+|TB|}{|OE|+|EF|} = \frac{13+2x}{1+x} = 2 + \frac{11}{1+x}$$

r is monotone decreasing, $r \leq r_B$. Else if T is on S_{CD}, the tour path is monotone increasing but $|OPT(P)|$ is always a little bit longer than $|OA|$. So, r is monotone increasing. $r \leq r_D$.

So, when T is on C_2, $r \leq \text{MAX}\{r_B, r_D\} = r_D = 18$.

Situation 2. T is on C_3. See Fig.6(b). As discussed in situation 1, we have $r \leq r_H < 33$.

Situation 3. T is on C_m ($m \geq 4$). See Fig.6(b). As discussed in situation 1 and the proof of theorem 1, we have $r < 24.75$.

All situations had been enumerated above, in term of the intersection point T. The proof is thus complete.

4.2 Case 2: Two Groups

In this situation, the evacuees are divided into two groups, and the communication between the two groups is always available. Both groups follow the same strategy, except that they leave in opposite directions at the first step. Once a group reaches the boundary of P, another group walks along the shortest path to it. The details of our strategy are described as follows, see Fig.7.

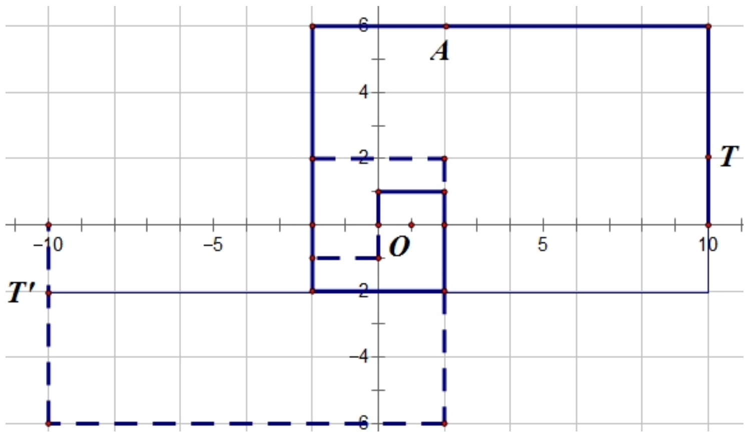


Fig. 7. The strategy TSEG

Two Groups Square Evacuation Strategy in Grid Network (TSEG)

Step 1: G_1 and G_2 leave from O in direction $+\overline{y}$ and $-\overline{y}$, respectively. They follow the tour path like that in strategy OESG until one of them reach the boundary of P.

Step 2: The group that does not arrive at any boundary of P stop and go to the intersection point between P and the path of the arrived group.

Theorem 4: In grid network, the evacuate ratio of TSEG is no more than 19.

Proof. In this case,

$$r = \frac{|SWI(P)|}{|OPT(P)|} = \frac{|S_{OT}| + |L_{\mathrm{TT}'}|}{|OPT(P)|}$$

As proved in theorem 2 and theorem 3, we have $r \le r(A) < 19$. See Fig.7.

The proof is thus complete.

5 Concluding Remarks

In this paper, we study the problem of finding efficient strategies such that the evacuees can leave from the affected area as soon as possible without boundary information. We first present efficient strategies with one or two groups and analyze the competitive ratios of them. In general plane, we give a 30.47-competitive strategy for one group evacuation and a 15.58-competetive strategy for two-group evacuation. In grid network, we give a 33-competitive strategy for one group evacuation and a 19-competive strategy for two-group evacuation.

There are some other related questions which may be interesting for further research. First, it is an interesting work to improve our competitive ratios. Second, it would be interesting to see how new strategies can be developed so as to efficiently solve the evacuation problem in concave polygon.

Acknowledgements. This work was supported in part by National Natural Science Foundation of China (No. 61173034) and the General Project of Liaoning Province Science and Research (No. L2012487).

References

1. Hoffmann, F., Icking, C., Klein, R., Kriegel, K.: The polygon exploration problem. SIAM J. Comput. 31(2), 577–600 (2002)
2. Chen, X., Zhan, F.B.: Agent-based simulation of evacuation strategies under different road network structures. Journal of the Operational Research Society 59, 25–33 (2008)
3. Lu, Q., George, B., Shekhar, S.: Capacity constrained routing algorithms for evacuation planning: A summary of results. In: Medeiros, C.B., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 291–307. Springer, Heidelberg (2005)
4. Shekhar, S., Yang, K., Gunturi, V.M.V., Manikonda, L., et al.: Experiences with evacuation route planning algorithms. International Journal of Geographical Information Science 26(12), 2253–2265 (2012)
5. Berman, P.: On-line searching and navigation. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms 1996. LNCS, vol. 1442, pp. 232–241. Springer, Heidelberg (1998)
6. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrum, S.: Collaborative multirobot exploration. In: Proceedings 2000 ICRA, Millennium Conference, IEEE International Conference on Robotics and Automation, Symposia Proceedings, vol. 1, pp. 476–481 (2000)
7. Xu, Y., Qin, L.: Strategies of groups evacuation from a convex region in the plane. In: Fellows, M., Tan, X., Zhu, B. (eds.) FAW-AAIM 2013. LNCS, vol. 7924, pp. 250–260. Springer, Heidelberg (2013)
8. Wei, Q., Wang, L.J., Jiang, B.: Tactics for evacuating from an affected area. Proceedings 2013 ICCCI, IJMLC 3(5), 435–439 (2013)
9. Miyazaki, S., Morimoto, N., Okabe, Y.: The online graph exploration problem on restricted graphs. IEICE, Trans. Inf. & Syst. E92-D(9), 1620–1627 (2009)

A Novel Adaptive Web Service Selection Algorithm Based on Ant Colony Optimization for Dynamic Web Service Composition

Denghui Wang, Hao Huang*, and Changsheng Xie

Wuhan National Laboratory for Optoelectronics/School of Computer Science and Technology, Huazhong University of Science & Technology, Wuhan 430074, China
wangdenghui1856@gmail.com, thao@hust.edu.cn

Abstract. Trust degree and quality of service are nonfunctional properties of component service. In dynamic web service composition, trust degree is variable. It can be changed according to execution logs. In this paper we establish a novel model of dynamic web service composition based on variable trust degree and quality of service. Then we propose one adaptive ant colony optimization algorithm to solve this multi-objective optimization problem. In the final, a case study shows that the proposed algorithm can find more Pareto solution than the traditional ant colony optimization algorithm. And the results also prove that our novel algorithm has higher efficiency than the traditional one.

Keywords: strust degree, Pareto solution, adaptive ant colony optimization algorithm.

1 Introduction

The purpose of web service composition is selecting functional units from different web service providers, and integrating them to meet customer complex requirement. In network there exist plenty of services that have the similar functional features and different nonfunctional features. These nonfunctional features include trust degree, quality of service (QoS), etc. QoS attitudes are provided for evaluating the quality of service. They are very important parameters of web service selection. However, due to excessive competition growing, service provider always exaggerated QoS values [16]. So we should consider the trust degree of component service. In dynamic web service composition, trust degree can be calculated through execution logs [15]. This paper establishes a novel web service composition model based on variable trust degree and QoS.

Ant colony optimization (ACO) is a kind of meta-heuristic search algorithm, it applies global optimization problem. The basic idea of ACO is to solve the search for finding a minimum cost food path in a graph [7]. ACO can not only solve the static single-objective optimization problem but also solve dynamic multi-objective optimization problems. While more and more improved ant colony

* Corresponding author.

algorithm are proposed in these years. In this paper we improve the traditional ACO. According to the variable trust degree we can adjust pheromone evaporation factor dynamically, and make the routing behavior of ants according to the dynamic change of environment.

The remainder of this paper is organized as follows: this second chapter focuses on new researches of global optimal web service composition, and introduces the application of ACO in other areas. The third chapter describes the details of novel web service composition model. The adaptive ant colony optimization (AACO) is proposed in fourth chapter. The fifth chapter gives the experiment that compares AACO with the traditional ACO and analyses the result. The sixth chapter gives conclusion.

2 Related Works

Recently most of researchers force on QoS-aware component service selection in service composition. Cardoso et al. [1] establishes a QoS parameter system including cost, response time, reliability, availability and reputation. Research of [2,3] use different mathematical methods to achieves QoS-aware component service selection in composite service. Besides QoS parameters, Trust degree is very important nonfunctional parameter. But there has been little reported application of trust degree. Li et al. [4] establishes the trust degree through user experience and record history.

The majority of existing web service composition optimal is based on the multiple constraints of QoS parameters. In [5,6] multi-restriction parameters of service QoS are transformed into single-goal function by weighted linear method. The exhausted calculation method and the heuristic calculation method are used for web service composition optimal. Exhaustive calculation method is poor scalability, computationally intensive. The performance of the heuristic calculation method is better than exhaustion calculation method. Zhang et al. [12] gives out genetic algorithm for service composition selection. Liu et al. [8] proposes a global optimizing and multi-objective algorithm based on multi-objection genetic algorithm. Besides genetic algorithm, ACO algorithm is applied to web service composition optimization [10]. While traditional ACO is widely used, but still keep some of its own inherent defects such as slow convergence. To overcome these drawbacks, many researchers have proposed improved schemes such as Huang et al. [9] proposes MMAS algorithm that avoid stagnation problems. Jiang et al. [11] estimates the theoretical approach of ACO convergence time based on mathematical model of Markov process.

3 Web Service Composition Model

3.1 User Requirements Model

R_{CW} denotes the user requirements set as R_{CW}=<input, output, precondition, result, constriction>, Where input and output are the set of input and output parameters of the composite web service; precondition specifies things that must be true before executing the composite web service. Result characterizes the physical side-effects that execution of the composite web service [14]. These four service description elements specify the user's functional requirements. Constriction specifies the user's nonfunctional requirement.

3.2 Abstract Web Service Model

AS denotes abstract web service as AS=<input, output, precondition, result>. AS is a set of web service instances that have similar functionality. AS consist of functional elements.

3.3 Web Service Instance Model

WS_i denotes a web service instance that consists of abstract service type and nonfunctional properties as WS_i=<AS, QoS, Trust degree>. CW is the collection of web service instances as CW=$< WS_1, WS_2, \cdots, WS_i >$.

3.4 QoS Evaluation Score Calculation Model

As we know QoS parameters include cost, response time, reliability, availability and reputation. Cost and response time is negative attributes. The value of negative attributes is smaller, the performance is better. So the evaluation score of negative QoS attributes is:

$$SQ_{neg} = \frac{q_{max} - q_{neg}}{q_{max} - q_{min}} . \tag{1}$$

In which q_n represents the value of negative QoS attribute. On the other hand, reliability, availability and reputation are positive attributes. The value of positive is bigger, the performance is better. So the evaluation score of positive attributes is:

$$SQ_{pos} = \frac{q_{pos} - q_{min}}{q_{max} - q_{min}} . \tag{2}$$

In which q_p is the value of positive QoS attributes. According to user requirement we can get weight value vector of every service instance as $\omega =< \omega_1, \omega_2, \cdots, \omega_n >$ [13]. Through weight value vector we can use (3) to get the QoS parameters evaluation score of web service instance i.

$$SQ_i = \sum_{n=1}^{N} \omega_n * SQ_{in}, (1 \leq n \leq N) . \tag{3}$$

3.5 Trust Degree Evaluation Model

We can calculate trust degree through execution logs. Every component service node can record the number of successful implement in execution logs. The trust degree will be changed in dynamic web service composition. So the evaluation of trust degree is:

$$TD_i(t) = \frac{N(ws_i)}{N(ws_{AP})} . \qquad (4)$$

In which, $TD_i(t)$ denotes the trust degree of web service instance i at the t moment. $N(ws_i)$ represents the successful execution number of web service instance i. $N(ws_{AP})$ is the total execution number of abstract service type.

3.6 Web Service Composition Model

There is a set of web service instances for each abstract service type on network. According to the constraint conditions of service composition which includes functional requirements and nonfunctional requirements (QoS, Trust degree), create a concrete workflow that stitches together the desired functionality from the existing service.

4 Adaptive Ant Colony Optimization Algorithm

The multi-objective web service composition selection problem can be concluded as finding shortest path P_{short} from component service instance set on directed graph. We show the directed graph G=<V,E> as Fig. 1.

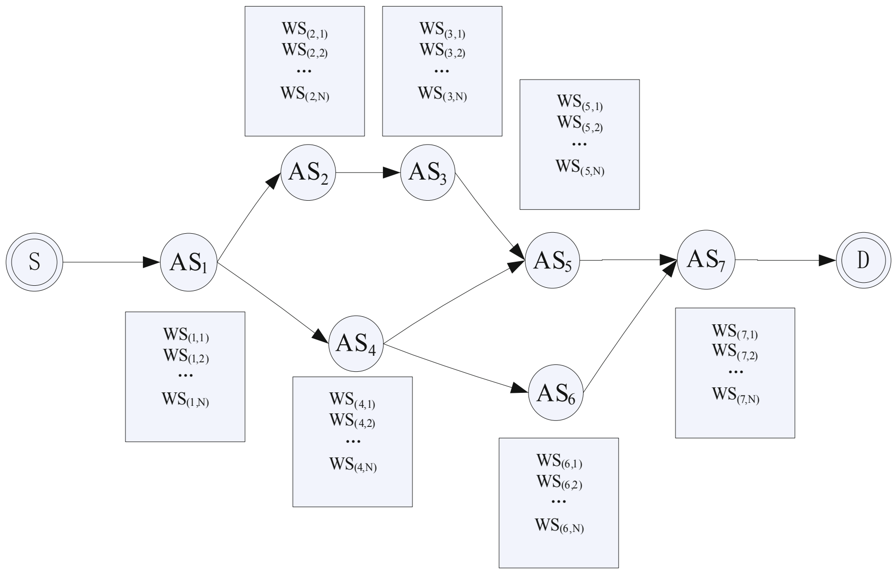


Fig. 1. The directed graph of workflow web service composition

V represents the set of all service instance node. E is the set of potential links between the web service instance nodes of the neighbored abstract service types.

S and D is virtual beginning node and target node. Based on the directed graph G, P=<S,D> represent a path from the beginning node S to the target node D.

4.1 Distance between Two Instances

The distance between the service instance i and the service instance j is:

$$d_{ij}(t) = \frac{1}{TD_i(t) * TD_j(t)} \ . \tag{5}$$

In which, $TD_i(t)$ is the trust degree of service instance i, $TD_j(t)$ is the trust degree of service instance j. The trust degree of service instance will be updated automatically after iteration.

4.2 Pheromone Update Rules

Pheromone in service instance node is initialized as:

$$\tau_{ij}(0) = (N_\delta + N_\xi) / \sum_{i=1}^{N_\delta} \sum_{j=1}^{N_\xi} d_{ij} \ . \tag{6}$$

Where N_δ is the service instance number of the abstract service type δ, N_ξ is the service number of the abstract service type ξ.

$$\tau_{ij}(t + N_{AP}) = \rho * \tau_{ij}(t) + (1 - \rho) * \tau_{ij}(0) \ . \tag{7}$$

In which N_{AP} is the abstract service type number of service composition. The ρ is the pheromone strength coefficient in ant colony algorithm. It is used to prevent to fall into premature and stagnation. But it is almost fixed. Therefore, in the proposed adaptive ant colony algorithm we adjust pheromone strength coefficient dynamically as:

$$\rho(t+1) = c * \rho(t) \ . \tag{8}$$

In which c is constant. Assumed that M is the total number of ants, after all ants finish iteration, pheromone will be updated according to global rule:

$$\tau_{ij}(t + M * N_{AP}) = \rho * \tau_{ij}(t) + (1 - \rho) * \triangle\tau_{ij}(t) \ . \tag{9}$$

In which, $\triangle\tau_{ij}(t)$ can be present as:

$$\triangle\, \tau_{ij}(t) = \sum_{k=1}^{M} \triangle\tau_{ij}^{k}(t) = \begin{cases} \sum_{k=1}^{M} \frac{Q}{LP(k)} \\ 0 \end{cases} \ . \tag{10}$$

In which, k denote the ant index. LP(k) represent the path of the ant k. The Q is constant.

4.3 State Transition Rules

The trust degree determines pheromone update process. But the constraint conditions of multi-objective web service composition include QoS requirement. So we use QoS evaluation score to represent heuristic value in AACO as:

$$\eta_{ij} = SQ_j \; . \tag{11}$$

At the t moment, one ant of service instance i find service instance j through rules below:

$$P_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{s \subset CN_i}[\tau_{is}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & j \in allowed \\ 0 & j \notin allowed \end{cases} \; . \tag{12}$$

In which, α is pheromone factor and β is expectation heuristic factor. CN_i is the collection of the web service instances which are the neighborhood of web service instance i.

4.4 Detail Steps of AACO Algorithm

The detail step of AACO is described as below:

1. Initiate the pheromone according to (6); set iteration counter $N = 0$; set the max iterations number N_{max}; set the counter of pheromone strength coefficient $k = 0$; set ant number is M; put all ants to the beginning node.
2. $N = N + 1$, if $N > N_{max}$, terminate the program.
3. $k = k + 1$, if $k = K$, set $k = 0$ and adjust pheromone strength coefficient according to (8).
4. Ant number plus 1.
5. Choose next node for this ant according to (12).
6. If the next node is not target node, turn back to step 5 and put this service instance node to taboo table of this ant, else go to step 7.
7. If the ant is not the last one, turn back to step 4, else go to step 8.
8. Calculate the trust degree of all service instance node and update pheromone.
9. Get the Pareto solution, turn back to step 2.

5 A Case Study

In this section we design an experiment to compare the performance of AACO with traditional ACO. We give two groups of different parameters for traditional ACO which are presented in Table 1.

We use these parameters to test four group data which present in Table 2. The represents the number of the abstract service type. The represents the web service instance number of each abstract service type. We assume that the number of ants is 20. Each group runs 100 and 200 iterations. The QoS evaluation score values of service instances are varying according to Gaussian distribution function. The constraint condition for the concrete workflow is CD=<SQ,TD>=<80,80>.

Table 1. Parameters of AACO and ACO

Algorithm	Parameter
ACO1	$\alpha = 1.0, \beta = 1.0, \rho = 0.8$
ACO2	$\alpha = 1.0, \beta = 1.0, \rho = 0.4$
AACO	$\alpha = 1.0, \beta = 1.0, \rho = 0.95 * \rho(t), \rho(0) = 0.9, K = 10$

Table 2. Amount of abstract service type and service instance

Group	μ	ν
1	10	10
2	20	20
3	40	20
4	40	40

The testing process is described as follow:

1. Every algorithm is run fifty times to calculate an average for one group data.
2. Get the found solution set P_{alg} that include non-dominated solutions and dominated solutions.
3. Delete the dominated solutions to get the Pareto solution set of every algorithm as P_{true}:
   (a) P_{ACO1}: Pareto solution which is got with ACO1.
   (b) P_{ACO2}: Pareto solution which is got with ACO2.
   (c) P_{AACO}: Pareto solution which is got with AACO.
4. Obtain the Pareto solution set of each group date as follow: $P = P_{ACO1} \vee P_{ACO2} \vee P_{AACO}$.
5. Calculate the percentage of the accurate rate as AP.
6. Calculate the percentage of the efficiency as EP.

The experiments results is shown in Table 3.

Table 3. Experiments results

Group	Iterations	Algorithm	P_{true}	P_{alg}	AP	P	EP
1	100	ACO1	18	18	100%	22	82%
		ACO2	22	23.2	95%	22	100%
		AACO	22	22	100%	22	100%
	200	ACO1	20.2	20.2	100%	22	92%
		ACO2	22	24.3	91%	22	100%
		AACO	22	22	100%	22	100%
2	100	ACO1	30	33.6	89%	38	79%
		ACO2	31	35.5	87%	38	82%
		AACO	33	36.7	90%	38	87%
	200	ACO1	32	34.1	94%	38	84%
		ACO2	35	38.8	90%	38	92%
		AACO	37	38.6	96%	38	97%

Table 3. (*Continued*)

Group	Iterations	Algorithm	P_{true}	P_{alg}	AP	P	EP
3	100	ACO1	41	51	80%	62	66%
		ACO2	44	57.9	76%	62	71%
		AACO	50	58.1	86%	62	81%
	200	ACO1	45	54.4	83%	62	73%
		ACO2	52	62.8	83%	62	84%
		AACO	56	64	88%	62	90%
4	100	ACO1	48	62.4	77%	79	61%
		ACO2	52	70.2	74%	79	66%
		AACO	60	72.7	83%	79	76%
	200	ACO1	50	65.6	76%	79	63%
		ACO2	59	79.1	75%	79	75%
		AACO	67	70	84%	79	85%

After analysising the data of Table 3, we can know that the accurate rate of AACO is better than ACO2 and the efficiency of AACO is better than ACO1 in Group1. We also find that the AACO show better performance than ACO1 and ACO2. In fact, AACO always get more Pareto solutions for all run times in Group2, Group3 and Group4.

6 Conclusion

This paper introduces trust degree and QoS parameters as nonfunctional property, establishes mathematic model of web service composition. And then an adaptive ant colony optimization algorithm is proposed to resolve the question of global multi-objective optimization with QoS and trust degree constraints. The experimental result shows that the proposed algorithm can find more Pareto solution in unit time. So the proposed algorithm has better performance that the traditional one.

Acknowledgments. Denghui Wang would like to extend sincere gratitude to corresponding author, Hao Huang, for his instructive advice and useful suggestions on this research. And we thank the anonymous reviewers for their valuable feedback and suggestions.

References

1. Cardoso, J., Miller, J., Sheth, A., Arnold, J.: Modeling Quality of Service for Workflows and Web Service Processes. Web Semantics Journal: Science, Services and Agents on the World Wide Web 1, 281–308 (2004)
2. Qu, Y., Lin, C., Wang, Y., Shan, Z.: QoS-aware Composite Service Selection in Grids. The Fifth International Conference on Grid and Cooperative Computing, pp. 458–465 (2006)

3. Jin, H., Chen, H., Lv, Z., Ning, X.: QoS Optimizing Model and Solving for Composite Service in CGSP Job Manager. Journal of Computer 28, 578–588 (2005)
4. Li, C., Cheng, B., Chen, J.: A Web Service Performance Evaluation Approach Based on Users Experience. In: IEEE International Conference on Web Service, pp. 734–735 (2011)
5. Liu, Y.T., Anne, H.H., Zeng, L.Z.: QoS Computation and Policing in Dynamic Web Services selection. In: Proc. WWW 2004, pp. 66–73. ACM, New York (2004)
6. Jorge, C., Amit, S., John, M.: Quality of Service for workflows and Web Service Processes. Journal of Web Semantics 1, 281–338 (2004)
7. Gao, C., Wang, J., Dong, Z.: Searching trust path model based on ant colony algorithm. Computer Engineering and Applications, 131–133 (2007)
8. Liu, S., Liu, Y., Jing, N., Tang, G., Tang, Y.: A Dynamic Web Services selection Strategy with QoS Global Optimization Based on Multi-objective Genetic Algorithm. In: Zhuge, H., Fox, G.C. (eds.) GCC 2005. LNCS, vol. 3795, pp. 84–89. Springer, Heidelberg (2005)
9. Han, H., Hao, Z., Wu, C., Yong, Q.: The convergence speed of ant colony optimization. Chinese Journal of Computers 30, 1344–1353 (2007)
10. Fang, Q., Peng, X., Liu, Q.: A Global QoS Optimizing Web Services Selection Algorithm based on MOACO for Dynamic Web Service Composition. In: International Forum on Information Technology and Applications, pp. 37–42 (2009)
11. Jiang, H., Mao, Z., Liu, X.: Research of software defect prediction model based on ACO-SVM. Chinese Journal of Computers 34, 1148–1154 (2011)
12. Zhang, C., Sen, S., Chen, J.: Genetic algorithm on Web services selection supporting QoS. Chinese Journal of Computers 29, 1029–1037 (2006)
13. Zhu, R., Wang, H., Feng, D.: Trustworthy services selection based on preference recommendation. Journal of Software 22, 853–864 (2011)
14. Dai, G., Wang, Y.: Trust-aware Component Service Selection Algorithm in Service Composition. In: International Conference on Frontier of Computer Science and Technology, pp. 613–618 (2009)
15. Wang, Y., Vassileva, J.: A review on trust and reputation for Web service selection. In: Proc. 27th International Conference on Distributed Computing Systems Workshops, pp. 25–32 (2007)
16. Li, Y., Zhou, M., Li, R., Cao, D.: Service Selection Approach Considering the Trustworthiness of QoS Data. Journal of Software 19, 2620–2627 (2008)

An Optimization VM Deployment for Maximizing Energy Utility in Cloud Environment

Jinhai Wang[1,2], Chuanhe Huang[1], Qin Liu[1], Kai He[1], Jing Wang[1], Peng Li[1], and Xiaohua Jia[3]

[1] Computer School, Wuhan University, Wuhan, China
[2] College of Computer, Xinjiang Vocational University, Urumqi, China
[3] Department of Computer Science, City University of Hong Kong, Hong Kong SAR
{wangjinhai,huangch,qinliu}@whu.edu.cn, csjia@cityu.edu.hk

Abstract. With rapid growth of the demand for computation power, which has led to establish plenty of large-scale data centers consuming enormous amount of electrical power. Energy consumption has become a critical problem. We propose an energy efficient multi-dimension resource allocation algorithm for virtualized Cloud datacenters that reduces energy costs and provides required Quality of Service (QoS). Our VM deployment algorithm achieves a good balance between energy and performance by minimizing the amount of provisioning servers as well as maximizing time sharing of VMs hosted on the same server. Energy saving is achieved by VM deployment, continuous consolidation according to current utilization of resources, workload demand and load states of computing nodes. Our scheme achieves a good balance between energy consumption and performance. Meanwhile, we adopt DPS (dynamic powering on/off servers) techniques to power on/off servers and buffer the change of workload, and also adjust consolidation threshold dynamically. The results show that our proposed strategies bring sustainable energy saving while ensuring reliable QoS.

Keywords: Cloud Computing, Energy Utility, Virtual Machine Deployment.

1 Introduction

Cloud computing[1] has become a consolidated paradigm for delivery of services through on-demand provisioning of virtualized resources, and users enjoy infrastructures, platforms, and software(applications) as a service in a pay-as-you-go manner from anywhere and at anytime. Studies have shown that energy waste is very huge because servers in many data centers are often severely under-utilized due to over-provisioning for the peak demand , which typically run at less than 30% of their full capacity[2]. The cloud model is expected to offer automatic scale up and down in response to load variation. It reduces hardware cost and saves electricity bills which contribute to the significant portion of the operational expenses in large data centers.

Workload fluctuation is a huge challenge not only in system elasticity but also in energy efficiency [3]. After the peak, there were plenty of resources left idle. Low utilization of servers is one of the most important factors of low energy utility for datacenters. To save energy, we should improve the scalability and power off some physical machines (PMs) according to dynamic workloads by adopting some certain mechanisms. Virtualization techniques such as Xen provide a mapping from VMs to physical resources elastically. The key issue is how to decide the mapping adaptively to meet the resource demands as well as minimize the number of PMs assigned without compromising QoS.

The aim of VM deployment is to improve resource utilization and reduce energy consumption while meeting QoS of users. To determine which components of critical workloads can be packaged together, we must carefully consider the combinations of different workloads under common physical suitability of the hosts. This is because different combinations will result in different energy utility and performance. The proposal in [4] produced plenty of hot spots which would incur more migrations to avoid SLA violations. Although the policy proposed in [5] considered multi-dimension VM deployment, the authors did not consider the impact of service time upon the deployment and migration.

To reduce energy consumption, we need to increase the workload per unit of energy can perform. Therefore, we use Energy Efficiency Rate to measure the extent of saving energy for a data center. This paper focuses on high energy utility resource allocation algorithm for virtualized datacenters. Our proposed algorithm is a greedy heuristic algorithm based on a novel utility we defined to implement the mapping from VMs to hosts to solve energy utility maximum ILP problem. Our policy considers not only resources efficiency and energy but also the service time of demands when deploying VMs. The key point of our algorithm is to achieve a good balance between energy and performance when minimizing the amount of provisioning servers as well as maximizing time sharing of VMs hosted on the same server. Meanwhile, we reduce the number of migrations and present more types of VM candidates while needing migration. The DPS techniques we use can dynamically improve energy efficiency of datacenter with dynamic thresholds. The experiment results show that there is significant improvement not only in energy saving but also in workload balancing and scalability comparing with existing methods such as single-objective approaches based on CPU utilization.

2 Related Work

Green Cloud computing is envisioned to achieve not only the efficient processing and utilization of a computing infrastructure but also to minimize energy consumption[6]. Existing energy-efficient resource allocation solutions proposed for various computing systems cannot be implemented for Green Cloud computing. They did not emphasize autonomic energy-aware resource management mechanisms exploiting VM resource allocation which is the main operating technology in Cloud Computing. [7] presented a method of VM consolidation management to dynamically reduce the number of nodes.

However, we consider multiple factors to consolidate VMs such as reducing migration is another consolidation during the phase of VM placement. In [8], the researchers present a novel pattern-driven application consolidation (PAC) system to achieve efficient resource sharing in virtualized cloud computing infrastructures by employing signal processing techniques. [9] proposed an algorithm that allowed multiple applications to share a single machine, and strived to maximize the total satisfied application demand, to minimize the number of application starts and stops, and to balance the load across machines.

Beloglazov et al. [4] proposed an energy-aware allocation heuristics provision data center resources to client applications in a way that improves energy utility of the data center, while delivering the negotiated Quality of Service (QoS). However, they proposed allocation algorithm the Modified Best Fit Decreasing (MBFD) algorithm only considers the CPU utilization, actually, because VMs requested by users are various, sometimes, requests are partial to high CPU, but sometimes partial to high memory; so that the physical nodes may increase because VMs need more memory to allocate when hosted on the server. More hosts may mean more live migration, which cause more migration cost.

L.-T. Lee et al. [10] proposed a dynamic resource management mechanism with energy saving, the authors presented a method of dynamic voltage scaling for dynamic adjustment of resources by inspecting CPU utilization in the cloud computing environment. However, they didn't consider the problem of the deployment of VM. Zhen Xiao, et al [5] introduced a "skewness" algorithm to measure to unevenness in multi-dimensional resource utilization of a server. However, they did not consider the effect of service time upon energy utility, which may also bring out the increase of migrations.

Comprehensively considering the aforementioned discussed studies, we propose a more efficient resource allocation algorithm for VM deployment and migrations. We use dynamic powering on/off servers technique (DPS) to further save energy. Our study is different from the existing work in which consider migration reduction when deploying VM due to maximizing time sharing of VMs, and effectively handle strict SLAs and do not depend on particular workloads and yet not require any knowledge about applications running in VMs.

3 Model Description and Problem Formulation

3.1 User Instance Model

In this paper, we present a time-slotted cloud system consisting of many networked servers. Each server may host multiple VMs. Each VM requires a set of resources including CPU, memory, and so on. Cloud providers offer different types of VMs with different QoS and pricing models that help them support various applications. We respond user requests using Amazon EC2-style VMs shown in Table 1. For these types of virtual instances (e.g., small, medium, large), they may be computation-intensive, memory-intensive or medium type. We focus on centralized scheduling strategies which involve two types of resource mapping: the mapping from application workloads to resource requirements through a global scheduler and the mapping from virtual resources to physical resources through local controllers. A local manager in local physical node is responsible for determining the amount of resources allocated to VMs that can guarantee application performance at minimum cost.

In our scheme, user demand set *D* is responded by VM instance set. User demand set is an *N*-dimension vector, and each element of *D* denotes a type of VM request during a time period. $D=(d_1, d_2, d_i, \ldots, d_N)$, $d_i(v_i, L_i)$, d_i denotes a type of VM request, $v_i=(cpu, memory, disk, network)_i$ is a resource set, $L_i=(l_i, t_i^s, t_i^e)$, user $i\in I$, *I* is a set of users, each user request represents one type of VM. Let l_i, t_i^s, t_i^e denote user service time, service start time and end time, respectively.

Table 1. Virtual machine instances

Amazon EC2 Cloud Platform--Amazon EC2 instance types					
Server size	Configuration			Cost/hr	$/core
Small	1 ECU	1.7GB RAM	160GB disk	$0.085	$0.085
Large	4 ECUs	7.5GB RAM	850GB disk	$0.34	$0.085
Med-Fast	5 ECUs	1.7GB RAM	350GB disk	$0.17	$0.034
XLarge	8 ECUs	15GB RAM	1.7TB disk	$0.68	$0.085
XLarge-Fast	20 ECUs	7GB RAM	1.7TB disk	$0.68	$0.034

3.2 Energy Model

For the computing nodes in data centers, CPU consumes the main part of energy comparing to other resource such as memory and disk storage. Hence, in this work we focus on managing its power consumption and energy utility. In addition, the CPU utilization is typically proportional to the overall system load. Recent studies [11, 12] have shown that the application of DVFS on the CPU results in almost linear power-to-frequency relationship for a server. Energy consumption is approximately $p=\delta CV^2 f$, where δ is an activity factor which is a constant for a certain server[6], *C* is the loading capacitance, *V* is the supply voltage, and *f* is the clock frequency. In the ideal case, the supply voltage and the clock frequency are related in such a way that $V \propto f^{\varphi}$ for some constant $\varphi > 0$.For ease of discussion, assume the datacenter is homogenous and let $V=af^{\varphi}$ (*a* is some constants). The total power of datacenter is

$$P_{all}=\sum_{j=1}^{M} p_j=\sum_{j=1}^{M}\delta_j C_L V_j^2 f_j = a^2\delta C_L \sum_{j=1}^{M} f_j^{2\varphi+1} \tag{1}$$

where *M* is the number of active servers. In order to guarantee the Quality of Services, the total provisioning frequency cannot be lower than a certain value. Assume $\Gamma=\sum_{j=1}^{M} f_j$ be capable of a given workload with certain SLA, in order to reduce P_{all}, we consolidate the VMs to reduce the number of active physical servers (APMs).

3.3 Definition of Energy Utility

Energy utility is an important metric for a cloud datacenter. To comprehensively every factor including each resource and service request time of users when deploying VMs, we give a novel definition for the energy utility of server as follow.

$$u_j = \frac{r_j}{p_j} \tag{2}$$

where p_j denotes the average power during all the time-slots performing all the loaded demand on server j, r_j is the total average resource utilization for a physical server during the same period. Either improving r_j or reducing p_j can improve energy utility of datacenters. The symbols r_j and p_j is defined as the following formula (3), (4). The variable $r_{j\tau s}$ denotes the usage percentage of resource τ of server j during time slot s. This definition indicates the total resource utilization of system and power consumption over the periods of VM performing. The variable t_{ijs} tells whether VM i runs on server j during time slot s, and I_j denotes all the users loaded on server j, t_{ij}^s and t_{ij}^e denote the start time of user i and the end time respectively.

$$r_j = \sum_{s=t_j^s+1}^{t_j^e} \sum_{\tau=1}^{k} r_{j\tau s} t_{ijs} \tag{3}$$

$$p_j = \frac{e_j}{t_j} = \frac{\sum_{s=t_j^s+1}^{t_j^e} p_j(s)}{\max\{t_{ij}^e, i \in I_j\} - \min\{t_{ij}^s, i \in I_j\}} \tag{4}$$

We define the utility of VM as in (5), which is the sum of resource percentage of every resource during the time slots of VM i. The energy utility of a user (VM) is defined as

$$u(v_i) = \sum_{\tau=1}^{k} \left(\frac{r_{i\tau}}{r_{j\tau}} l_i \right) \Big/ p_j \tag{5}$$

where p_j is the average power during VM i running on server j, which can be calculated by (4).

3.4 Problem Formulation

To reduce energy consumption, we need to increase the workload per unit of energy can perform. Therefore, we use Energy Efficiency Rate to measure the extent of saving energy for a datacenter. The workload is the processing capacity allocated to VMs. Assume the number of VMs is N, and need M servers to respond. R denotes a set of resource types (i.e. CPU, memory, disk and network). The formulation of VM deployment problem is described as the following maximization that is also an optimization **ILP formulation**.

The objective

$$\max \sum_{j=1}^{M} \frac{\sum_{i=1}^{N} a_{ij} u_i}{p_j} \tag{6}$$

Subject to

$$\sum_{j=1}^{M} a_{ij} c_j(t) \geq \sum_{i=1}^{N} c_i(t)(\forall t) \tag{7}$$

$$\sum_{i=1}^{N} a_{ij} c_i(\tau) \leq c_j(\tau)(\forall t, \forall \tau \in R) \tag{8}$$

$$\sum_{j=1}^{M} a_{ij} = 1, \forall i \text{ and } a_{ij} \in \{0,1\} \tag{9}$$

where a_{ij} is an integer variable in the ILP that can take value of 0 or 1. The value of 1 indicates that the VM i is mapped to server j during the time slot s, and the value of 0 indicates that it is not loaded during the time slot s. The output of the ILP is set of values a_{ij} that denotes which VM is placed on which server. In (7), (8) c_i and c_j denote the capacity of server and VM respectively. Constraint (7) guarantees the capacity of provisioning resources to handle user demands. Constraint (8) guarantees the capacity of each server is not less than the capacity of all the VMs loaded on it. The last constraint indicates a VM can only be loaded on a server.

4 Proposed Strategy

It is not difficult to see that the aforementioned problem is a NP-hard problem, and, it is intractable to find optimal solution when the system size is large. In this work, we adopt greedy heuristics to find a feasible near-optimal solution by the definition of energy utility in section 3.3.

4.1 Illustration of Deployment

To achieve high degree of VM consolidation, we must find a viable deployment using the minimum number of nodes. Fig.1 shows the motivation of our scheme. Suppose that there are 2 PMs (PM_1 and PM_2) each one with 12 available ECUs and four VMs (VM_1, VM_2, VM_3 *and* VM_4) with different execution time (6 hours, 6 hour, 3.5 hour, and 3 hours respectively), and require different ECUs execution (4ECUs, 5ECUs,5ECUs,4ECUs). There are multiple packing schemes, but only Fig.1(d) is optimal, neither of others like Fig.1(b) and (c) is viable because both of them exist potential migration. Due to the intuition, we propose our heuristic deployment algorithm as follows.

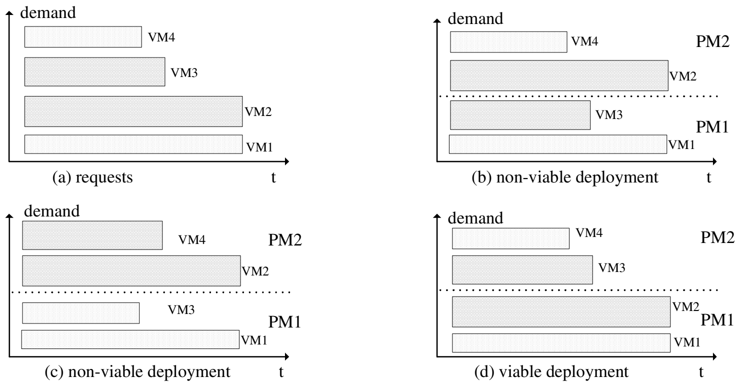


Fig. 1. Non-viable and viable deployment

Algorithm 1:EUD

```
        INPUT: hostList, vmList
        OUTPUT: allocation
1   vmList.sortDecreasingUtility();        //based on formula (5)
2   foreach vm in vmList do
3       maxUtility ← MIN
4       allocatedHost ← null
5       foreach host in hostList do
6           if host has enough resource for vm then
7               utility ← evaluate(host,vm)     //based on formula (2)
8               if utility >maxUtility then
9                   allocatedHost ← host
10                  maxUtility ← utility
11          if allocatedHost ≠ null then
12              allocate vm to allocatedHost;
13  return allocation
```

4.2 VM Deployment

Due to the aforementioned definition of energy utility and above illustration, we propose our VM deployment algorithm. First, we give the following lemma to demonstrate the deployment principle.

Lemma: assume delivering resource $\Gamma = \sum_{j=1}^{M} f_j (f_j > 0)$ is competent to a certain SLA workload, here M is the smallest number of provisioning servers but can meet user requests, the total energy consumption $P_{all} = a^2 \delta C_L \sum_{j=1}^{M} f_j^{2\varphi+1}$ is the smallest when $f_1 = f_2 = f_3 = \cdots = f_m = \Gamma / M$.

***Proof*:** This is an extremum condition. We can easily prove it by the method of Lagrange multipliers. Let $\varphi(f_1, f_2, \cdots, f_m) = \sum_{j=1}^{M} f_j - \Gamma = 0$, resolve the followed equation (10)

$$\begin{cases} \partial P_{all} / \partial f_1 + \lambda \partial \varphi(f_1, f_2, \cdots, f_m) / \partial f_1 = 0 \\ \partial P_{all} / \partial f_2 + \lambda \partial \varphi(f_1, f_2, \cdots, f_m) / \partial f_2 = 0 \\ \vdots \\ \partial P_{all} / \partial f_m + \lambda \partial \varphi(f_1, f_2, \cdots, f_m) / \partial f_m = 0 \\ \varphi(f_1, f_2, \cdots, f_m) = 0 \end{cases} \tag{10}$$

we can gain the solution $f_1 = f_2 = \cdots = f_m = \Gamma / M$ and λ. However, it is an ideal condition in generally. If $f_j \neq \Gamma / M$, to gain a closed extremum, we can reduce P_{all} by reducing the difference between f_i and Γ / M .

Due to the lemma, when keeping the total frequency constant for a certain workload, the more balanced the frequency of the active servers, the lower of the energy consumption. Furthermore, we consider every type resource and time sharing of VMs hosted on the same server for the energy utility definition. Hence, it is easy to achieve workload balance when deploying VMs. The VM deployment processing is that mapping every VM in vmList to a server and makes the server energy utility biggest. The details of deployment are shown in the following Energy Utility VM Deployment algorithm (EUD) shown in Algorithm 1. Firstly, we sort the VMs of vmList by decreasing order of energy utility (calculated by formula (5)). Secondly, we evaluate the utility of each server by formula (2), then allocate the VM to the server and make the server with biggest energy utility. The deployment is over until the vmList is empty.

4.3 VM Live Migration

VM live migration can be divided in two parts: the first part is workload balancing, and the second part is VM consolidation. To guarantee system performance and save energy, we set double thresholds of processor utilization. At the same time, based on the double thresholds, we divide the status of a server into three statuses: cold, warm, and hot. To save energy and guarantee performance, we should make every server work in warm status as much as possible.

$$\begin{cases} \{v \mid v \in V_j, u_j - \sum_{v_i \in v} u(v_i) < T_{upper}, \ | u'_j - u_{average} | \to \min\} & if \ u_j > T_{upper} \\ V_j & if \ u_j < T_{lower} \\ \varnothing & otherwise \end{cases} \tag{11}$$

The migration rule is shown in (11), T_{upper} and T_{lower} are the upper threshold and lower threshold for migration that will be given through experiments. On one hand, to avoid SLA violation, some VMs will be migrated to other servers to mitigate current load when CPU utilization of a server violates the upper threshold. On the contrary, all the VMs must be migrated to other servers when CPU utilization violates the lower threshold, then set the server in corresponding mode (such as sleep mode or powering off) to further save energy. The algorithm of VM consolidation migration is similar to algorithm 1, so we no longer give the details. Furthermore, to void SLA violations, workload balancing is performed preferentially when both of them occur at the same time. The main aim of workload balancing is to make the workload difference among the active servers smallest.

4.4 Dual-Threshold Server On/Off Policy

To further save energy, there are usually two server-level energy consumption optimal techniques which are DVFS (Dynamic Voltage Frequency Scaling) and DPS (Dynamic Powering On/Off Servers). DVFS is used to reduce energy consumption by adjusting CPU working frequency. However, a server consumes about 60%-70% energy of peak even though zero workload loaded on it [4], so it is necessary to power off some idle APMs to save energy further.

DPS technique dynamically powers on or off servers to save energy due to system workload. Because user demands fluctuates and is not predicted, as well as powering on/off servers both spends time, all these will influence system performance and incur additional energy overhead. Therefore, DPS should reserve a part of idle servers to buffer dynamic workload. At the same time, system should adopt appropriate strategies to determine the opportunity of powering on/off servers. We use the method proposed in [13] as shown in formula (12) to count flow in real time during a period.

$$\sigma(t) = \frac{1}{M^*}\int_0^t G_\mu^c(t-\mu)\lambda(\mu)d\mu \tag{12}$$

Where, $\sigma(t)$ denotes the average utility of system over time duration *t*. $G_\mu^c(t) = 1 - G_\mu(t) = P(s(\mu) > t)$. $s(\mu)$, $\lambda(\mu)$ indicates performing time and arrival rate of arrival tasks at the moment μ. M^* is the number of servers gained by VM placement algorithm. We adjust server state (On/Idle/Off) to resolve the problem of opportunity of powering on/off server and server reserve by dual-threshold policy. We adopt dual-threshold policy to control servers powering on/off. The dual-threshold policy We adopt is similar with the method in [14]. We do not immediately power on servers when workload exceeds the capacity of powering on servers, until workload accumulate to a certain rate β_1(threshold β_1). Moreover, we also do not immediately power off servers when some servers are in idle, until idle server amount accumulates to a certain rate β_2(threshold β_2). β_1 and β_2 can be calculated by formula (13) and (14) respectively.

$$\beta_1 \approx \frac{1}{M^*}\log_{\sigma(t)}\frac{M^*(1+\beta_1)\mu}{\vartheta}\left(T - \frac{1}{M^*(1+\beta_1)\mu(1-\sigma(t))}\right) \tag{13}$$

$$\beta_2 \approx \frac{1}{M^*}\log_{\sigma(t)}\frac{M^*(1+\beta_2)\mu}{\theta}\left(T - \frac{1}{M^*(1+\beta_2)\mu(1-\sigma(t))}\right) \tag{14}$$

where T is the upper limit user defined. ϑ and θ indicate startup rate and shutdown rate respectively(reciprocal of startup time and shutdown time respectively).

5 Evaluation

We conduct simulations and experiments from several metrics to evaluate the performance of our algorithm comparing with the two algorithms MBFD[4] and "skewness" [5] which have been commented in section 2 related work in this paper.

5.1 Simulations and Results

We evaluate the performance of our algorithm using trace driven simulation. We collected the traces from servers and desktop computers in our university, including high performance computing data center, mail servers, the central DNS server, desktops in our department. We post-processed the traces based on days and use random sampling and linear combination of the data sets to generate the workloads needed. We achieve simulations using CloudSim toolkit[15]. It supports workload variable modeling of on-demand virtualized resource, application management and energy-aware simulations. Each PM is modeled to have one CPU core with the performance equivalent to 1000, 2000 or 4000 MIPS, 10 GB of RAM and 1 TB of storage. Each VM requires one CPU core with 250, 500, 750, 1000 or 1500 MIPS, 128 MB or 256MB of RAM and 1 GB of storage.

Thresholds decide the extent of VM consolidation, too low if it is, will VM be hosted on even more servers and more migration be operated, more energy also will be consumed. However, too high if it is, will more SLA violation occur although energy be saved.

We define that an SLA violation occurs when a given VM cannot get the amount of resource that are requested. This can happen in cases when VMs sharing the same host require that cannot be provided due to the consolidation. This metric shows the level by which the QoS requirements negotiated between the resource provider and consumers are violated due to the energy-aware resource management. Fig.2 shows that energy consumption decreases as CPU utilization increases. That is because higher threshold make the higher level of VM consolidation. However, SLA violation also increases. This is due to that a higher utilization threshold allows more aggressive consolidation of VMs and the increased risk of SLA violations as shown in Fig 3.

We first evaluate the effect of the various thresholds used in VM deploying and migrating. We simulate a system with 100 PMs and 1200 VMs that are selected form the trace randomly. We first conduct simulations to evaluate the effect of different deployment algorithm and various thresholds used in our algorithm. As shown in Fig.4, Simulations show our algorithm with a certain improvement in the number of active physical machines (APMs) than "skewness" when not considering migration factor. However, our algorithm has much less migrations (as shown in Fig.8) when taking service time of VM into account. Meanwhile, our scheme has much better improvement than MBFD in VM placement.

To guarantee server performance avoiding its utilization too high and save energy when a server is under-utilized, we set a double thresholds policy. We migrate some VMs when CPU utilization violates the upper threshold, On the other hand, we migrate all the VMs on the server to others when its CPU utilization violates the lower threshold. Fig.5 shows the effect of different threshold interval (i.e. UL(90%, 30%) denotes upper threshold 90% and lower 30% respectively). It is obvious that lower threshold play a key role to reduce the number of APMs when workload demand is in the period of declining, especially. In our scheme, we adopt the threshold interval UL(90%, 30%) as our migration metric.

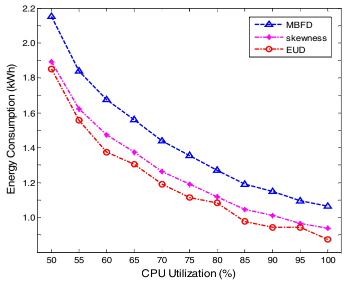


Fig. 2. CPU threshold with energy consumption

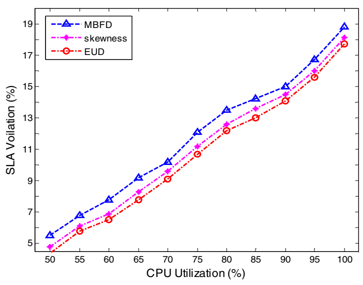


Fig. 3. CPU threshold with SLA

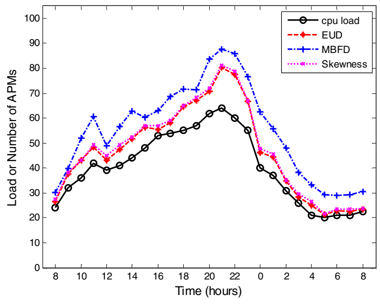


Fig. 4. Comparison of different policies

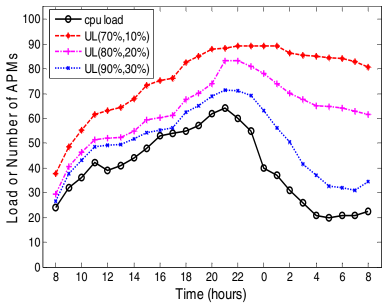


Fig. 5. impact of thresholds upon APMs

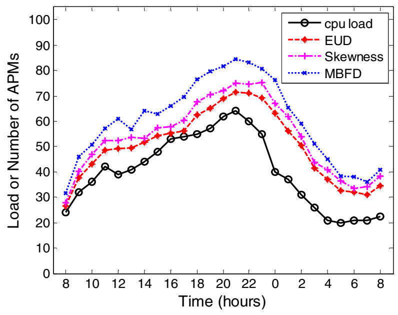


Fig. 6. Comparison of the number of APMs

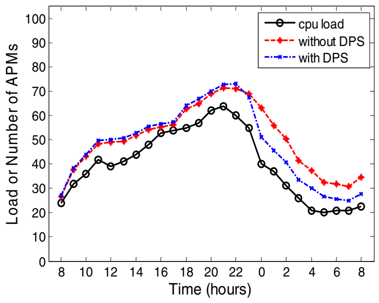


Fig. 7. The number of APMs with DPS

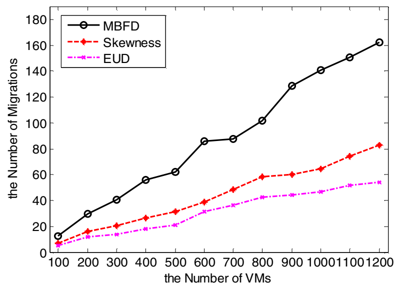


Fig. 8. The number of average migration

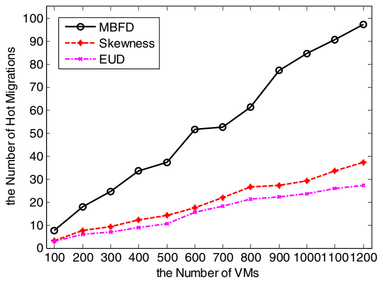


Fig. 9. The number of hot migration

We evaluate the scalability of our algorithm in deployment and migration. The number of active servers changes as requests fluctuating. Fig.6 demonstrates that the number of APMs coincides with workloads much better adopting our policy under the same threshold interval constraints. That is to say our algorithm is more effective than the two algorithms MBFD and "skewness" in VM placement. Fig.7 shows DPS technique effectiveness. Dynamic consolidation threshold can improve utilization of resources and energy, especially when workloads decline. That is because we adopt dynamic thresholds mechanism, when workload demands decline and system utility also declines, that we improve cold threshold can further consolidate VMs. Our policies decrease not only the number of active servers but also the migration. The advantage of our policies in migration decreasing has been clearly shown in Fig.8 when the scale of VMs is very big. Specially, when the density of computing intensive services is more, the CPU-based greedy algorithm MBFD has a very big scale of migration under thresholds constraining. However, our algorithm has a good effectiveness because which balances the loads among the active servers as much as possible. Furthermore, hot migration incurs more SLA violations. We also evaluate the number of hot migrations. The results show the reduction of our scheme in this aspect in Fig.9.

5.2 Experiments

Our experiments are conducted using a group of 20 Dell PowerEdge blade servers with Intel Xeon E5620 CPU and 32GB of RAM. The servers run Xen-3.3 and Linux 2.6.18. We periodically read load statistics using the *xenstat* library (same as what *xentop* does). The servers are connected over a Gigabit Ethernet to a group of three NFS storage servers where our VM Scheduler runs.

We implement Xen-based VMs and evaluate our approach using workloads based on TPC-H because TPC-H provides 22 representative queries of business decision support systems, which involve the processing of large volumes of data with a high degree of complexity. Based on these queries, we construct synthetic workloads by varying demands of different types of resources (i.e. we subject the VMs to different CPU load by adjusting the request rates of users). The parameter β_1 and β_2 actively take the value in our DPS strategy based on states of system load. When $\sigma(t)$ is lower, we increase the lower threshold to further consulate the VMs, and vice versa.

5.3 Experiment Results

To quantify the energy saving, we measured the electricity consumption under various fluctuated workloads including the traces of high performance computing datacenter (such as CPU-intensive services, memory-intensive services and so on), mail servers and TPC-H using different algorithm for two hours. We find that a fully utilized blade server consumes about 236 Watts and an idle blade server 152 Watts. The experiment results in Fig.10 have shown that our scheme consumes about 6.25 kWh, saves the most energy. However, "skewness" and MBFD consume 6.77 kWh and 7.46kWh respectively. Furthermore, if adopt a Non Power-Aware (NPA) policy, 14.32 kWh energy would be consumed. Compared with the three policies, our policy saves energy about 7.68%, 16.22% and 129.12% in our experiment respectively.

SLA is another metric of evaluation. We define that an SLA violation occurs when a given VM cannot get the amount of resources that are requested, or the service fails to finish until deadline. In the above experiment, Fig.11 shows that our scheme has an evident improvement. Compared with MBFD, it is due to two aspects: i) Our algorithm reduces the number of imbalance servers such as some servers allocated more CPU-intensive VMs but some servers allocated more memory-intensive VMs and so on. ii) The imbalance of workloads deployment increases the number of migrations. Both increase not only energy consumption but also SLA violation. However, compared with "skewness", the increase of the number of migrations incurs more SLA violation than our algorithm because of migration increase the cost of energy and time, especially under the bad network conditions.

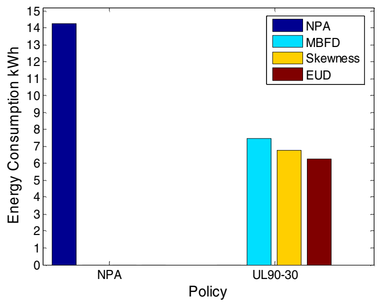


Fig. 10. Comparison of energy

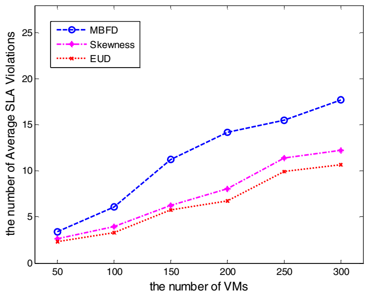


Fig. 11. Comparison of SLA violation

One concern about the use of live migration is its impact upon performance. We conduct experiments and investigate the impact. Fig.12 shows that it is very small when system utility is low, however, it would incur much higher SLA violations when system utility is more than 90%. The reason is migrations would consume the cost of resources, time and also energy. Especially when system workload is higher, hot migration incurs more SLA violations. Therefore, we must consider reducing potential migrations when deploying VMs. Our proposed policy just does it.

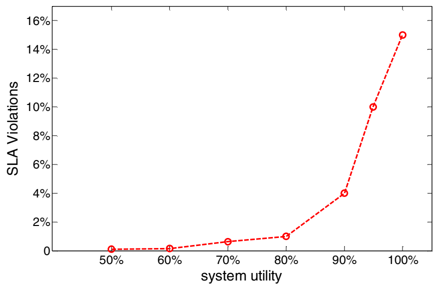


Fig. 12. Impact of migration upon SLA

6 Conclusion

In this paper, our proposed algorithm demonstrates its high efficiency. It not only can take good use of every type resources, but also achieve a good balance between energy and performance when minimizing the number of APMs. And there is a remarkable reduction of VM migration. PMs have a better consolidation and performance improving. DPS techniques we adopted avoid powering on/off server frequently according current workload. As reducing the energy waste, we yet reduce SLA violations and improve Quality of Service by setting the dual thresholds. The experiment results show the good performance of our policy.

Acknowledgements. This work is supported by the National Science Foundation of China (No. 61373040, No.61173137, No. 61103217), the Ph.D. Programs Foundation of Ministry of Education of China(20120141110073), Key Project of Natural Science Foundation of Hubei Province (No. 2010CDA004).

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.: A view of cloud computing. Communications of the ACM 53, 50–58 (2010)
2. Sargeant, P., Managing, V.: Data Centre Transformation: How Mature is Your IT?, Presentation by Managing VP, Gartner (2010)
3. Chen, F., Grundy, J., Schneider, J.-G., Yang, Y., He, Q.: Automated Analysis of Performance and Energy Consumption for Cloud Applications. In: Proceedings of 2014 ACM/SPEC International Conference on Performance Engineering (2014)
4. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28, 755–768 (2012)
5. Xiao, Z., Song, W., Chen, Q.: Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. IEEE Transactions on Parallel and Distributed Systems 24, 1107–1117 (2013)

6. Qingjia, H., Sen, S., Jian, L., Peng, X., Kai, S., Xiao, H.: Enhanced Energy-Efficient Scheduling for Parallel Applications in Cloud. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 781–786 (2012)
7. Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G., Lawall, J.: Entropy: A consolidation manager for clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 41–50 (2009)
8. Zhenhuan, G., Xiaohui, G.: PAC: Pattern-driven Application Consolidation for Efficient Cloud Computing. In: 2010 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 24–33 (2010)
9. Tang, C., Steinder, M., Spreitzer, M., Pacifici, G.: A scalable application placement controller for enterprise data centers. In: Proceedings of the 16th International Conference on World Wide Web, pp. 331–340 (2007)
10. Lee, L.-T., Liu, K.-Y., Huang, H.-Y., Tseng, C.-Y.: A Dynamic Resource Management with Energy Saving Mechanism for Supporting Cloud Computing. International Journal of Grid and Distributed Computing 6, 67–76 (2013)
11. Verma, A., Ahuja, P., Neogi, A.: pMapper: power and migration cost aware application placement in virtualized systems. In: Issarny, V., Schantz, R. (eds.) Middleware 2008. LNCS, vol. 5346, pp. 243–264. Springer, Heidelberg (2008)
12. Metri, G., Srinivasaraghavan, S., Weisong, S., Brockmeyer, M.: Experimental Analysis of Application Specific Energy Efficiency of Data Centers with Heterogeneous Servers. In: IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 786–793 (2012)
13. Jennings, O.B., Mandelbaum, A., Massey, W.A., Whitt, W.: Server staffing to meet time-varying demand. Management Science 42, 1383–1394 (1996)
14. Ke, J.-C.: Optimal NT policies for M/G/1 system with a startup and unreliable server. Computers & Industrial Engineering 50, 248–262 (2006)
15. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41, 23–50 (2011)

Performance Evaluation of Light-Weighted Virtualization for PaaS in Clouds

Xuehai Tang[1], Zhang Zhang[1], Min Wang[2], Yifang Wang[3], Qingqing Feng[2], and Jizhong Han[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences
tangxuehai@iie.ac.cn, zhang_zhang@live.com, hanjizhong@iie.ac.cn
[2] Institute of Computing Technology, Chinese Academy of Sciences
wangmin@nelmail.iie.ac.cn, jenny2008.1@gmail.com
[3] Hunan University
wangyifangce@gmail.com

Abstract. The use of traditional virtualization technologies in Platform as a Service(PaaS) has been almost absent due to their inherent performance overhead. However, with the rapid development of light-weighted virtualization techniques, such as OpenVZ, Docker, Lmctfy and ZeroVM, they begin to be widely used in PaaS because of the possibility of obtaining a very low overhead comparable to the near-native performance of a bare server. In this work, we analyze these techniques and conduct a number of experiments in order to perform in-depth evaluations of light-weighted virtualization techniques for PaaS in clouds. We compare them in the proposed EIS(Efficiency, Isolation, Speed) framework. As far as we know, this paper is the first to propose an unified testing framework EIS to get a in-depth quantified analysis for Docker, Lmctfy, ZeroVM and as well as KVM, which is a representative of the mainstream hypervisor-based virtualization systems used today.

Keywords: Light-weighted Virtualization, Docker, Lmtfy, ZeroVM, KVM.

1 Introduction

In the era of cloud computing, the virtualization technology, as its corner stone, begins to develop rapidly with this hot trend, and there appears many more mature virtualization technology (such as Xen[15], KVM[7] and VMWare[14], etc.). Currently mainstream virtualization can be divided into three categories: full virtualization, paravirtualization and light-weighted virtualization. For the full virtualization and the paravirtualization, there is an additional heavy-weighted virtualization layer on top of the operating system architecture, which is responsible for emulating physical devices for virtual machines. With the help of some hardware support and some modifications or optimizations of the operating systems, the virtualized host has the ability to run many virtual machines with totally different operating systems. The main advantages of para and full virtualization are the software/hardware independence, isolation, flexibility and security. For these benefits, the para and full virtualization technologies are widely used in the cloud data center nowadays.

However, besides these advantages, there exists a fatal flaw for the para and full virtualization. That is, the large performance overhead for the additional virtulization layer in the host operation system. For this reason, these virtulizaion technologies are barely used in the field of PaaS. That's why the light-weighted virtulizaion comes into being, such Lmctfy[10], Docker[4], ZeroVM[16]] and so on. The light-weighted virtualization, only a kernel module is introduced into the host operating system, which does not need to emulate physical devices. Here, the virtual machine is actually a container with a group of processes running inside. The host operating system is directly responsible for the isolation of different virtual machines. In theory, the light-weighted virtulization can achieve the performance of bare physical sever with little system overhead.

With the advent of the era of big data, so the cloud computing platform presents a lot of classes of applications. In order to integrate the management of cloud applications and cluster resources, so the multiplexing cluster appeared that can improve resource utilization. The multiplexing cluster achieved efficient deployment and operation of applications on PaaS, but the applications on PaaS need a running environment of high resource utilization, certain isolation and rapid deployment. The traditional virtualization technology cannot meet this scenario while light-weighted virtualization is suited. So the light-weighted virtualization technologies are widely used in this case.

Consider the following scenario in PaaS:

Resource Sharing and Isolation. In the PaaS, the resources are generally controled by a resource scheduler to assign to different users/tasks of a job, such as Omega[20], Yarn [22], Mesos[18] and BAE[2]. The resource scheduler allows different users to share resources of cluster. There are three main resources which are CPU resources, memory resources and I/O resources. The resource requirements for each task is different and varied, if you want to improve the utilization of resources in the cluster as a whole, you need a scheduler which can be fine-grained allocation and management of resources. Therefore, there must be a mechanism for sharing and isolation to resource.

Fast Flexible Resource Management. While a platform run a variety of jobs, these jobs may have a different load with the change of time, because the jobs have particularity logic and uncertainty load, the changes of load may lead to changes in demand for resources. In order to deal with this scenario, the resource manager typically needs to dynamic distribute and recover resources for a number of jobs to achieve higher resource utilization. This requires the underlying resource management mechanism which can quickly respond to the changing of requirements in real time.

Custom Operating Environment. Cluster resources are usually shared by many jobs, these jobs may require different software packages/libraries and the operating system kernel and configuration. Even a lot of jobs can share some of the packages or the system kernel configuration, but it is difficult to ensure that they do not affect each other. Thus, a virtualization technology is extremely necessary to create a different runtime environment.

In the above scenarios, the light-weighted virtualization technologies are needed for the sharing and isolation of resource without loss of performance. Therefore, we use a series of benchmark to measure the current mainstream of light-weighted virtualization.

In this paper, we provide a reference for the selection of virtualization technologies in PaaS by compared this light-weighted virtualizations. In Section II, from analysis of light-weighted virtualization technologies, we know that its mainly consist of the control and isolation of resource. The control of resource has many technologies, such as UBS, fair CPU scheduling, Disk Quotas and I/O scheduling, cgroups and memory segments mechanism; the isolation of resource mainly has the mechanism of namespaces and sandbox. In Section III, from experimental results, we conclude that Lmctfy and Docker have better performances, more rapid speeds in deployment but weaker isolations; OpenVZ has a higher performance efficiency and better isolation, but with poor speed in deployment; KVM has the strongest isolation but the poorest performance in both efficiency and speed.

2 Light-Weighted Virtualization

Resource virtualization is implemented by the operating system software on the intermediate layer, which provides a abstraction of multiple virtual resource. In general, the virtual resources are called a virtual machine(VM)and can be seen as isolated execution contexts. There are many kinds of virtualization technologies, such as currently popular hypervisor-based virtualization, container-based virtualization and Unix process of abstraction.

There hypervisor-based virtualization, in its most common form (hosted virtualization), consists of a virtual machine monitor (VMM) on top of a host OS that provides a full abstraction of VM. The VM is a completely independent of the host operating system and other processes running VM environment.

Container-based virtualization, namely the operating system level virtualization technology, partitions the physical machines resource, creating multiple isolated user-space instances. Figure 1 shows container-based virtualization and Hypervisor-based virtualization at the different of the architecture. Hypervisor-based virtualization is an abstract comprehensive client operating system, and container-based virtualization works at the operation system level, providing abstractions directly for the guest processes.

Unix process of abstraction is not a virtual physical resources, but rather a virtual application. The application is compiled into the VM(virtual machine) and completely transparent to the host system layer and the hardware layer.

Currently, the mainstream of light-weighted virtualization is mainly divided into two categories: One is based on the operating system, namely container technology(such as OpenVZ, Docker and Lmctfy), The other is Unix process of abstraction(such as ZeroVM).

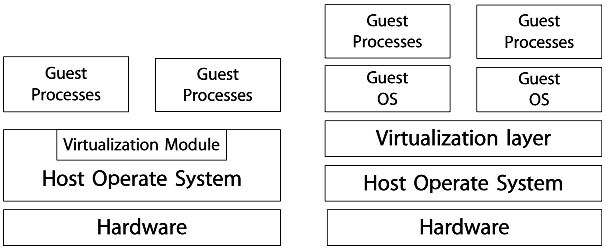


Fig. 1. Comparison of container-based and Hypervisor-based virtualization

2.1 OpenVZ

OpenVZ uses User Beancounters (UBS), fair CPU scheduling, Disk Quotas and I/O scheduling to achieve resource management. The UBS achieves a set of limits and guarantees controlled per-container done through control parameters. In this way, we can restrict memory usage and various in-kernel objects such as IPC shared memory segments and network buffers. The OpenVZ CPU scheduler is implemented in two levels to promote a fair scheduling among containers. The first level decides which container will get attention from the processor at some instant of time. The second level performs the scheduling of internal processes of the container according to priorities. The Disk Quotas can set up standard UNIX per-user and per-group disk limits for containers. Finily, a similar approach of CPU scheduling is used for I/O.

OpenVZ is based on kernel namespaces [11] technology for isolation of container resource. PID namespace provides a separate PID environment and an independent and unique process ID for each container process. IPC namespace make that each container has its own independent IPC objects, such as memory segments, semaphores and message. The PID namespace and IPC namespace can create a separate space, and different processes are not visible and cannot communicate with each other. Network namespace provides a completely independent network protocol stack for a process, Including network device interface. UTS namespace is a group of identifier returned by uname. The UTS namespace and Network namespace can make a separate virtual host name and cyberspace environment, just as the network as a stand-alone host. OpenVZ creates a container that has independent PID, IPC, FS, Network, UTS space for a process through these namespaces.

2.2 Docker

Docker is based on LXC (Linux container) [9]to implement, which has been made open since March 2013. The basic architecture of Docker is shown in Figure 2. The LXC achieves isolation and control of container resources, and The storage driver achieves copy-on-write of the file system and image management. The LXC controls of the container resource though Cgroups[3](control groups) which is supported by Linux kernel. The Cgroups can be limited and recorded a container which used physical resources(such as cpu, memory), which is make up of a number of subsystems, and a subsystem is a resource controller. As the OpenVZ, The LXC also uses namespaces to provide resource isolation among all containers.

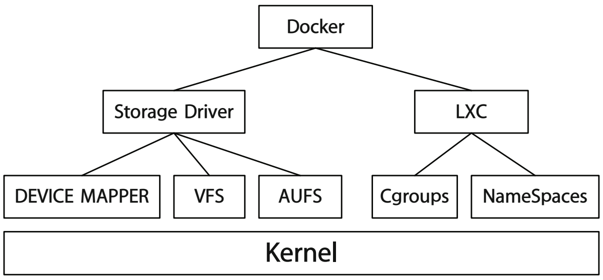


Fig. 2. Docker Architecture

Storage driver mechanism of Docker is the basis for copy-on-write of the file system and image management. Currently the storage driver main supports three file systems, namely AUFS[1], VFS and DEVICEMAPPER. The Aufs is a similar to Unionfs, which using the union mount can add a file system that supported reading and writing on a read-only file system. Docker called these file system as a layer, and read-only layer called a image. The principle achieved copy-on-write of the file system and image management.

2.3 Lmctfy

Lmctfy has been made open since September 2013 by Google. It can control and isolate resources of a process by a container, the resource such as CPU, memory and I/O bandwidth. The basic architecture of Lmctfy is shown in figure 3. Lmctfy consists of two layers, the CL1 and the CL2. The CL1 is responsible for the creation and maintenance of containers, and supporting strategies of the LC2. The CL2 is responsible for the setting of resources strategies. Currently, the CL1 only provides the isolation of resources, and the CL2 is still in the development stage.

As LXC, Lmctfy also uses the mechanism of Cgroups and namespace to control and isolate resources used by a container. Although the underlying implementation of Lmctfy is similar to LXC, but the LXC only corresponds to The CL1 of Lmctfy. The CL1 implement the abstraction of the underlying container. The CL2, through Resource Policy, will realize load awareness, so it will ensure the efficient utilization of resources and the high availability of the system. The Resource Policy will have the mechanism of priority and latency. lmctfy will manage all cgroup details to honor the priority and latency requirements for each task.

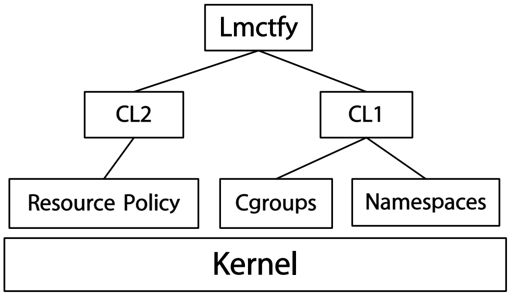


Fig. 3. Lmctfy Architecture

2.4 ZeroVM

ZeroVM has been made open since 2013. It is implemented based on the Google Native Client project. ZeroVM provides an abstraction of the lighted-weighted virtual machines by compiled under the C99 environment based on the APIs of the POSIX system. ZeroVM treats the I/O in the POSIX/UNIX as a file, sush as the input data as a standard input (STDIN), the log data as the standard error (STDERR) and the output data as a standard output (STDOUT).

ZeroVM implements the resource isolation by the security sandbox technology of Nacl[24] project. Nacl is known to have strong security isolation, which is based on isolated the untrusted code on a host system. Nacl accomplished strong isolation by using two layers of the sandbox mechanism, such as inner-sandbox and Outer-sandbox. The inner-sandbox uses static analysis to detect security defects in untrusted x86 code. Previously, such analysis has been challenging for arbitrary x86 code due to such practices as self-modifying code and overlapping instructions. In addition, the inner-sandbox further uses x86 segmented memory to constrain both data and instruction memory references. The Outer-sandbox is the second layer protection to defense untrusted code, by controlling the process of the system call to implement resource isolation and vulnerability defense. ZeroVM provides a safe running environment to applications with two layers, the inner-sandbox and Outer-sandbox.

3 Experiment

This section studies the performance in EIS (efficiency, isolation, and speed) of light-weighted virtualization and hypervisor-based virtualization. We performed several experiments with the current Linux container-based virtualization, such as OpenVZ, Docker and Lmctfy. We chose KVM as the representative of hypervisor-based virtualization because it is considered one of the most mature and efficient implementations of this kind of virtualization. The experiments are based on the existing benchmark, while ZeroVM need to re-design and build the procedures, so this experiment cannot compare the performance of ZeroVM

Table 1. Exprimental evironment

	OS Kernel	CPU	Memory	Disk
Evironment	Ubuntu-12.10	Intel Xeon E5-2640v2 x2	8G DDR3 1600LV X8	1 TB

Table 2. Exprimental benchmark

Test Object	Benchmark
Efficiency(CPU)	LINPACK benchmark
Efficiency(Memory)	STREAM benchmark
Efficiency(Disk)	IOzeone benchmark
Efficiency(Network)	NetPIPE benchmark
Isolation	Isolation Benchmark Suite Apache Benchmark
Speed	Definition benchmark

with others. The experimental environment and benchmark is shown in Table 2 and Table 3.

3.1 Efficiency

Computing Performance. To evaluate the computing performance, we choosed the LINPACK benchmark[8]. It simulations a compute-intensive task by a series of fortran subroutines that analyzes and solves linear equations. The LINPACK benchmark runs over a single processor and its results can be used to estimate the performance of a computer in the execution of CPU-intensive tasks. We ran LINPACK for matrices of order 5000 in all light-weighted virtualization technologies and compare them with KVM.

The results are shown in Figure 3, almost all light-weighted virtualizations obtained the performance similar to native. We believe that it is due to the fact that there are no influences of the different CPU schedulers when a single CPU-intensive process is run in a single processor. The results also show that KVM was not able to achieve the same performance, presenting an average overhead of 3.4%, because the Hypervisor need to convert cpu command.

Memory Performance. The memory performance was evaluated with STREAM benchmark[13], a simply program that measures the sustained memory bandwidth. It performs four type of vector operations, such as Add, Copy, Scale and Triad, using the data sets much larger than the cache memory available in the computing environment, which reduces the waiting time for cache misses and avoid memory reuse.

The results shown in Figure 4, all light-weighted virtualizations and native systems present similar performance, regardless of the vector operation. This is due to the fact that light-weighted virtualizations have the ability to return unused memory to the host and other containers, enabling better use of memory. The worst results were observed in KVM, which presented an average overhead of

4.1% when compared to the native system, this overhead is cause by hypervisor-based virtualization layer which perform memory access instructions translated, resulting in loss of performance.

Disk Performance. The experiment uses IOzone benchmark[5] to evaluate the disk I/O performance of light-weighted virtualizations. The IOzone benchmark can generate and measure a variety of file operations and access patterns (such as Initial Write, Read, Re-Read and Rewrite). The experiment ran the benchmark with a file size of 10GB and 4KB record size.

The results are shown in Figure 5, Lmctfy and native had a similar result because the Lmctfy and native share file system. Docker and OpenVZ had an almost similar result, but when you read, the Docker presented an average overhead of 4.1% when compared to the native system, and OpenVZ presented 67%. The result is shown that they have independent file system, and the AUFS and blkio subsystem of Cgroups has small overhead than the Disk Quotas and I/O scheduling. The worst result was observed in KVM for all I/O operations due to the hypervisor-based virtualization layer requires drivers to support disk I/O, and these drivers cannot achieve high performance.

Network Performance. The network performance was evaluated with the NetPIPE(Network Protocol Independent Performance Evaluator) benchmark[12]. The NetPIPE is a tool for measurement of network performance under a variety of conditions. It performs simple tests such as ping-pong, sending messages of increasing size between two processe through a network. The message sizes are chosen and sent at regular intervals to simulate disturbances and provide a complete test of the communication system. Each data point involves many ping-pong tests to provide accurate time measurements, allowing the calculation of latencies.

Figure 3 shows the comparison of the network bandwidth in each light-weighted virtualization. The Lmctfy obtained similar performance to the native, followed by Docker and OpenVZ. While Lmctfy does not implement virtualized network devices, both OpenVZ and LXC implement network namespaces that provides an entire network subsystem. We did not configure a real network adapter in OpenVZ system, as described in Section II, because it would reduce the scalability due the limited number of network adapter that normally exist in host machine. The worst result was observed in KVM, because the KVM network performance degradation is caused by the extra complexity of transmit and receive packets.

3.2 Isolation

To measure and evaluate the isolation of OpenVZ, Docker, Lmctfy and KVM, the experiment selects Isolation Benchmark Suite (IBS) [6] that demonstrates how much a virtualization can limit the impact of a guest with other guest running on a single host machine. This benchmark consists of six stress testing

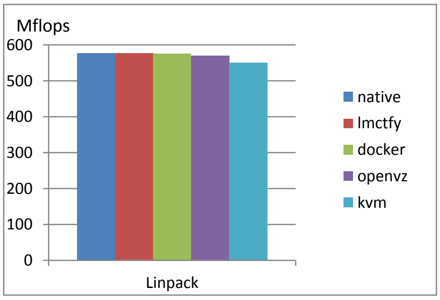


Fig. 4. CPU performance

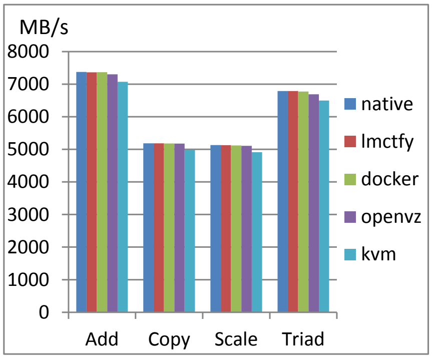


Fig. 5. Memory performance

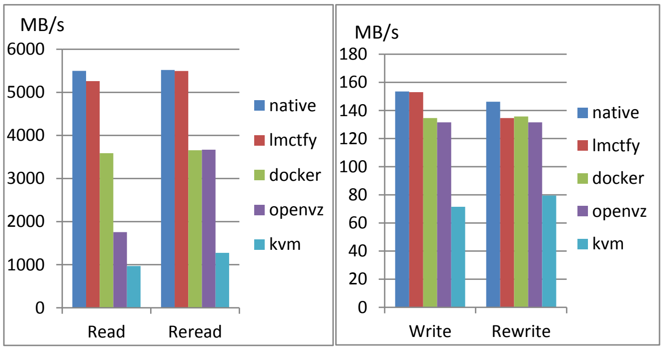


Fig. 6. Disk performance

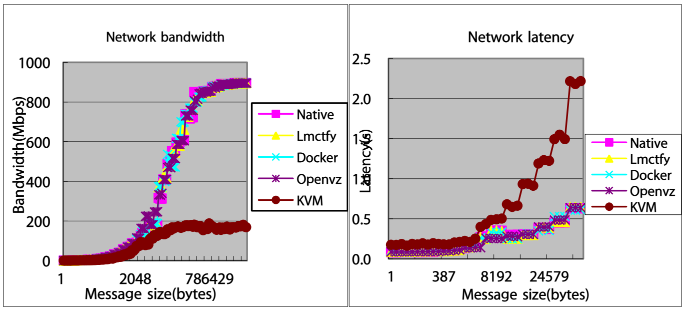


Fig. 7. Network performance

procedures which are CPU tests, memory test, fork bomb test, disk test and two network intensive tests (send and receive). For quantitative analysis of the isolation, the experiment used an Apache Benchmark to simulate application. In order to achieve the experiment, we built four virtual machines (A, B, C, D) on a single physical machine, and the virtual machine A ran the stress and the other virtual machines (B, C, D) ran Apache Server. When the virtual machine (A) in the absence of the stress, we got the T1 in virtual machine(B, C, D), which is the Apache Benchmark. Then, when virtual machine (A) ran the stress, we got the T2. Based on the value (T2-T1)/T1, we quantitative analyzed the isolation of the virtualization.

The results are shown in Table 4, CPU Stress has no effect on all virtualizations, which shows the VCPU Affinity and Cgroups of CPU subsystem can provide well CPU isolation. However, all others resources when stressed had some impact in other virtual machines (B, C, D). As described in Section II, all virtual instances in container-based virtualization shares a single operating system kernel. Hence, we supposed that while the kernel needs to handle instruction calls from the stressed guest (A), it is unable to handle instruction calls from the other guest (B, C, D). This behavior could have influenced the performance for all virtualizations while the memory, disk and network tests were performed. Fork Bomb is a classic test that loops creating new child processes until there are no resources available. The results indicated that Lmctfy and Docker have a security failure, due to the impossibility to limit the number of processes by Cgroups. As results, the isolation of KVM is better because kvm virtual machine and the host does not share the operating system.

3.3 Speed

The speed is light-weighed virtualizations can quickly build a virtual machine, which is a runtime environment to an application of PaaS. To measure and evaluate the speed, we designed of a Definition Benchmark program which is server/client architecture. The client simulated an application of PaaS. The server built a rpc server, and record the time of T1 that is the time of cre-

Table 3. Isolation Performance. DNR means that application was not able to run.

	Lmctfy	Docker	OpenVZ	KVM
CPU stress	0	0	0	0
Memory stress	53.3%	43.3%	20.1%	4.2%
Disk stress	34.4%	34.7%	36.8%	20.9%
Network Receivee	9.1%	6.7%	5.1%	3.0%
Network Sender	6.7%	6.3%	2.1%	0
Fork Bomb	DNR	DNR	0	0

Table 4. Speed Performance

	Lmctfy	Docker	OpenVZ	KVM
Time-consumings	0.202	0.189	13.197	21.588

ation a virtual machine, and got the time of T2 that is the time that the rpc server received a response of the client. We evaluated the performance of speed by the value of (T2-T1).

The results are shown in Table 5, Lcmtfy and Docker could create a virtual machine in 0.1 seconds level, due to they are implemented based on the Cgroups and namespaces which have been added to the system kernel. OpenVZ needed 10 seconds level to run an application in a virtual machine, due to OpenVZ need to copy and load the image to memory. Kvm running an applications in the virtual machine needed 20 seconds level, due to the kvm virtual machine supports independent system kernel, so this time is mainly consumed in coping the image to memory and loading system initialization.

To comprehensive analysis and evaluated the performance of these light-weighted virtualizations, we designed some formulas to calculate their efficiency, isolation, and speed. For the efficiency, according to the feature of application, we can know CPU as important as memory, disk and network. For the isolation, the memory was set to 50% because of it is an important factor to programs, and the fork bomb is a classic test so the proportion accounted for 20 %(DNR regarded as 100), and the other is set to 10%. For the speed, we get a time that is spent on creating a virtual machine, and then compare them with the minimum of time. This specific formula are shown in Table 5.

According to the experimental results and these formulas, we can know that Lmctfy and Docker have better efficient resource utilization and rapid large-scale deployment and weaker isolation; OpenVZ has the relatively better of efficiency

Table 5. Evaluate Formula

Formula
V(Effiency)= (cpu, memory, disk, network)LVT/Native*25%
V(Isolation)=100%-{(cpu, disk, network)*10% + (memory)*%50+(1-(fork))*%20}
V(Speed)= min{creating_time}/ (LVT){creating_time}

Table 6. Compare Light-weighted Virtualization Performance

	Efficiency(%)	Isolation(%)	Speed(%)
Lmctfy	97.5	49.5	93.6
Docker	95.0	55.8	100
OpenVZ	88.5	85.3	1.5
KVM	65.0	94.7	0.9

and isolation but the poor speed; kvm has the stronger isolation but the poorest of efficiency and speed. The Specific performance are shown in Table 7(the higher of the score and the better of the performance).

4 Related Works

There are already many papers studing the performance overhead of virtualization technology, but which rarely concern the performance of light-weighted virtualization technology for PaaS in cloud. Regola and Ducom[19] evaluated the performance of KVM, Xen and OpenVZ , and they used the NPB benchmark (OpenMP and MPI) and micro-benchmark for network and disk. Again, all virtualization systems obtained near-native performance for CPU intensive benchmarks. The OpenVZ is outstanding for I/O intensive benchmark. But they did not evaluate the isolation and speed of these virtualizations.

Walters et al.[17] evaluated the performance of VMware Server, Xen and OpenVZ. They also used the NPB benchmark (OpenMP and MPI) and micro-benchmark for network and disk. In their experiments, Xen and OpenVZ achieved near-native performance for the CPU intensive benchmarks, but OpenVZ outperformed Xen for the network intensive ones. However, they did not evaluate the isolation and speed.

Soltes et al.[21] presented the design and implementation of Linux-VServer and compared it with Xen, they used a benchmark for database server and micro-benchmark to measure the performance of cpu, disk and network. The experimental results shown that Linux-VServer provides comparable support for isolation and superior performance than Xen. The representation of this paper is not strong because of they only selected two virtualization technologies. They also did not evaluate the speed of these virtualizations.

Miguel et al.[23] analyzed and evaluated the Lxc, OpenVZ, Vserver and xen in the performance of high performance computing environments. They measured the performance of CPU, memory, disk, network and overhead in high performance environments. In their experiments, the performance of Lxc and Vserver are close to the native, and the isolation of OpenVZ and Vserver are better stronger. However, they did not evaluate the speed of these virtualization technology.

We believe that our work is complementary to the works presented in this section. we evaluated the efficiency, isolation and speed of the current mainstream of light-weighted virtualization technologies on PaaS environment. This article is the first work to evaluate the speed of light-weighted virtualization, and the feature is particularly important for PaaS in cloud. In addition, the paper is the first work to provide a EIS framework which can compare the performances of these light-weighted virtualization technologies.

5 Conclusion and Future Work

This paper provides a reference for the selection of virtualization technologies in PaaS by compared this light-weighted virtualizations. we analyzed these technologies in the control and isolation of resource, and evaluated these performance in EIS.

From Analysis of these virtualizations technologies, we know that its consist of the control and isolation of resource. From experimental results, we conclude that Lmctfy and Docker have better performances, more rapid speeds in deployment but weaker isolations; OpenVZ has a higher performance efficiency and better isolation, but with poor speed in deployment; KVM has the strongest isolation but the poorest performance in both efficiency and speed.

For future work, we plan to study the performance of light-weighted virtualization technology in the Cluster scheduling system. For example, we will measure the performance of light-weighted virtualizations for computing frameworks(MapReduce, Tez, Spark and storm)in Yarn or Mesos.

Acknowledgment. This research is supported in part by National High Technology Research and Development Program of China (863 Program) under grant 2012AA01A401, and Strategic Priority Research Program of the Chinese Academy of Sciences under grant XDA06030200.

References

1. AUFS, `http://aufs.sourceforge.net/` (Online; accessed April 8, 2014)
2. BEA, `http://developer.baidu.com/` (Online; accessed April 2, 2014)
3. Cgroups, `https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt` (Online; accessed April 7, 2014)
4. Docker, `https://www.docker.io/` (Online; accessed March 27, 2014)
5. IOZone Benchmark, `http://www.iozone.org` (Online; accessed April 10, 2014)
6. Isolation Benchmark, `http://web2.clarkson.edu/class/cs644/isolation` (Online; accessed April 12, 2014)
7. KVM, `http://www.linux-kvm.org` (Online; accessed March 11, 2014)
8. LINPACK Benchmark, `http://www.netlib.org/benchmark`, (Online; accessed April 10, 2014)
9. Linux container, `https://linuxcontainers.org/` (Online; accessed April 7, 2014)
10. Lmctfy, `http://www.linuxplumbersconf.org/2013/ocw/system/presentations/1239/original/lmctfy` (Online; accessed March 25, 2014)
11. Namespaces, `http://lwn.net/Articles/531114/` (Online; accessed April 2, 2014)
12. NetPIPE Benchmark, `http://www.scl.ameslab.gov/netpipe` (Online; accessed April 11, 2014)

13. STREAM Benchmark, http://www.cs.virginia.edu/stream/ (Online; accessed April 10, 2014)
14. VMWare, http://www.vmware.com (Online; accessed March 19, 2014)
15. Xen, http://www.xen.org (Online; accessed March 9, 2014)
16. Zerovm, http://zerovm.org/ (Online; accessed March 29, 2014)
17. Chaudhary, V., Cha, M., Walters, J., Guercio, S., Gallo, S.: A comparison of virtualization technologies for hpc. In: 22nd International Conference on Advanced Information Networking and Applications, AINA 2008, pp. 861–868. IEEE (2008)
18. Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R., Shenker, S., Stoica, I.: Mesos: A platform for fine-grained resource sharing in the data center. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, pp. 22–22 (2011)
19. Regola, N., Ducom, J.C.: Recommendations for virtualization technologies in high performance computing. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 409–416 (2010)
20. Sievers, F., Wilm, A., Dineen, D., Gibson, T.J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., et al.: Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. Molecular systems biology 7(1) (2011)
21. Soltesz, S., Pötzl, H., Fiuczynski, M.E., Bavier, A., Peterson, L.: Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In: ACM SIGOPS Operating Systems Review, vol. 41, pp. 275–287. ACM (2007)
22. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al.: Apache hadoop yarn: Yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing, p. 5. ACM (2013)
23. Xavier, M.G., Neves, M.V., Rossi, F.D., Ferreto, T.C., Lange, T., De Rose, C.A.: Performance evaluation of container-based virtualization for high performance computing environments. In: 2013 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 233–240 (2013)
24. Yee, B., Sehr, D., Dardyk, G., Chen, J.B., Muth, R., Ormandy, T., Okasaka, S., Narula, N., Fullagar, N.: Native client: A sandbox for portable, untrusted x86 native code. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 79–93. IEEE (2009)

An Access Control Scheme with Direct Cloud-Aided Attribute Revocation Using Version Key

Jiaoli Shi[1,2], Chuanhe Huang[1,⋆], Jing Wang[1], Kai He[1], and Jinhai Wang[1]

[1] Computer School, Wuhan University, Wuhan 430072, Hubei Province, P.R. China
[2] School of Information Science and Technology, Jiujiang University, Jiujiang 332005, Jiangxi Province, P.R. China
{shijiaoli,huangch,WJing,hekai_copper,wangjinhai}@whu.edu.cn

Abstract. Cloud storage allows owners to host their data in the cloud, and provides users with online access anywhere and anytime. With CP-ABE, data owners are allowed to specify policy autonomously, which can realize fine-grained access control. However, some important problems have not been yet effectively solved: 1) Low efficiency for attribute revocation. 2) High computational cost on encryption and decryption. Even if *direct revocation* has been proposed for a user's attributes, all ciphertexts with revoked attributes have to be re-encrypted. In this paper, we propose an access control scheme using version key to realize efficient *direct cloud-aided attribute revocation* without updating other user's key or re-encrypting ciphertexts. Revocation of a user's attributes just needs to update his own private key and version key stored in a cloud server, and most of decrypting work is transferred to the cloud. Moreover, we compare our scheme with two other schemes (DAC-MACS and HUR). The comparison shows a good trade-off between computation cost and storage overhead. Our simulation indicates that our scheme spends less time on a user's attribute revocation.

Keywords: Ciphertext-Policy Attribute-Based Encryption, Direct Cloud-Aided Revocation, Version Key, Cloud-Aided Decryption.

1 Introduction

Recent advancement of cloud storage technology enables people to easily share their data with others. Because cloud storage servers and data owners belong to different trust domains, the cloud storage server is not trustable. Data owners are worried about three issues: 1) Attackers illegally access sensitive data. 2) Some servers allow attackers to access sensitive data without permission. 3) The unauthenticated access of valid users is applied to sensitive data.

Attribute-Based Encryption (ABE) can achieve fine-grained access control in untrusted servers. According to the access control policy attached to the key and

⋆ Corresponding author.

the ciphertext, ABE has two branches: Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). Because owners define policies in ciphertexts, and CP-ABE is more suitable than KP-ABE for access control in cloud.

Nevertheless, some important problems have not yet been effectively solved. 1) Efficiency for attributes revocation is low. The change of user identity triggers revocation of one or more attributes. His attribute key then should be revoked. Meanwhile, those users who hold revoked attribute should update their private key, and those ciphertexts with revoked attributes should be re-encrypted. 2) Computational cost on encryption and decryption is high. The large amount of computational cost on encryption and decryption can be a lot of consumption of user's resource.

Considering problems discussed above, we present an access control scheme in which the computational cost on encryption and decryption is to be lower, and design a flexible efficient attribute revocation method that can achieve both *forward security* and *backward security* as well.

The main contributions of this work can be summarized as follows:

1) We propose an access control scheme which can realize fine-gained access control for multi-authority cloud storage systems.
2) We design a *direct cloud-aided attribute revocation* method in which a user's attribute-set version key pair is designed. When a user's attribute should be revoked, an authority just needs to update his private key and attribute-set version key stored in cloud server. In this case, an authority can revoke the user's attributes directly without updating other users' keys or re-encrypting ciphertexts encrypted with the revoked attribute.
3) We put forward a *cloud-aided decryption* method by which users are able to deliver most of computational work to cloud storage server, and do pairing operation only once, reducing the computational cost of decryption on each user.

2 Related Work

Access control methods up to date include CP-ABE[1], [2], DAC-MACS[3], [4], [5], HASBE[6], [7], [8], and so on. A number of works used CP-ABE to realize fine-grained access control for outsourced data [9], [10], [11]. However, the efficiency of computational cost and efficient revocation are open problem.

Yang et al.[3]proposed DAC-MACS (Data Access Control for Multi-Authority Cloud Storage), an effective and secure data access control scheme with efficient decryption and revocation. The scheme solved *key escrow problem.* However, users were required to transfer their private keys to cloud for generating a decryption token. The reveal of private keys increase the risk of security.

Hur et al.[12] put forward a novel CP-ABE scheme in which Two-party computation protocol was executed between Key Generation Center (KGC) and Data Storage Center (DSC) to solve *key escrow problem.* Fine-grained user revocation could be done by *proxy encryption* because of the selective attribute group key distribution on top of the ABE. However, the scheme incurred two shortcomings: 1) The request of attribute revocation should not be issued by user. 2) The addition of user or authority would result in all users' keys updating and ciphertexts re-encrypting.

Li et al.[13]established a patient-centric framework and a suite of mechanisms for data access control to PHRs (*Personal Health Record*). The users in the system were divided into multiple security domains. ABE as cryptographic primitives was applied. A set of key-generation rules and encryption rules were defined. *Forward security* and *backward security* were assured by Hash chain technology and a time-related ticket. However, the work just applied to PHR environment.

Bobba et al.[6]came up with CP-ASBE (*Ciphertext Policy Attribute Set Based Encryption*) scheme in which user attributes were organized into a recursive set and users were allowed to impose dynamic constraints on how those attributes might be combined to satisfy a policy. The main contribution was due to selectively allowing users to combine attributes from multiple sets within a given key while still preventing collusion. Wan et al.[7]formulated HASBE (*Hierarchical Attributes-set Based Encryption*) scheme based on Bobba's CP-ASBE. The main contribution was construction of a full scheme.

Wu et al.[14]devised a novel MCP-ABE (*Multi-message Ciphertext Policy Attribute-Based Encryption*) technique, and designed an access control scheme for sharing scalable media based on data consumers' attributes. The scheme constructed a key graph which matched users' access privileges, encrypts media units with corresponding keys, and then encrypted the key graph with MCP-ABE. Only those data consumers with the required user attributes could decrypt the encryption of the key (sub) graph and then decrypt the encrypted media units.

Müller et al.[15]explored the expression of access policy. The monotonic syntax tree was proposed in which all inner nodes were labeled with either $\vee$ or $\wedge$ and the leaves represented either Boolean variables or the constant values $\perp$. The main contribution of this paper was the formation of access tree which could be used to format the rules of key-generation, encryption and decryption.

Nuttapong et al.[16]presented the first *Hybrid Revocable ABE* scheme that allowed owners to select on-the-fly when encrypting whether to use either direct or indirect revocation mode. It combined best advantages from both direct and indirect methods.

3 Our Access Control Scheme

3.1 Problem Statement

We focus on solving the efficiency problems of attribute revocation and decryption on each user.

A. The efficient revocation of a user's attribute
It is extremely hard to revoke a user's attribute in an efficient way. Considering the following situation, when an executive of a company resigned, it would be

unrealistic that all the data that he could access would have to be re-encrypted immediately. He cannot decrypt the new ciphertexts that require the revoked executive attributes to decrypt (*Backward Security*). Meanwhile, the new executive can decrypt the previously encrypted ciphertexts if he has proper attributes (*Forward Security*). To ensure *forward security* and *backward security*, two revoking methods can be used: *direct revocation* and *indirect revocation. Direct revocation* does revocation directly: the owner specifies the revocation list while encrypting, and re-encrypts the ciphertexts on cloud which have been encrypted with the revoked attributes. *Indirect revocation* updates keys for non-revoked users and re-encrypts the ciphertexts encrypted with the revoked attributes. However, it does not meet the practical application that *direct revocation* and *indirect revocation* all result in re-encrypted ciphertexts en-crypted with the revoked attribute. Thus, attribute revocation is an operation that requires efficient computation in an access control scheme to revoke a user's attributes immediately.

B. The efficient decryption on each user
It is another efficiency problem that the computational cost on the decryption of users requires efficient computation in an access control scheme. Nowadays, users might carry their lightweight terminals to operate their data in present cloud storage systems. For this reason, the computational cost of decryption on users must be reduced as much as possible.

We address the aforementioned problems in its entirety in this paper.

3.2 Threat Model

We assume that: 1) Certification is full trusted. 2) Authority and cloud storage servers are semi-trusted.

Collusion attack. If user A and user B collude, they might decrypt the ciphertext by combining their attributes while either of them cannot decrypt the sensitive data.

Data leakage. Unauthorized users illegally access the sensitive data, so can cloud storage servers.

Revocation failure. Users revoked might decrypt the new ciphertexts encrypted with the revoked attributes. Users newly joined cannot decrypt the previous ciphertexts even if they have proper attributes.

3.3 System Model

The system model (shown as in Fig. 1) consists of five entities: ***CA*** (a global certificate authority), ***AAs*** (the attribute authorities), ***owners*** (the data owners), ***users*** (the data consumers) and ***cloud*** (the cloud server).

CA. It is a trusted certificate authority. All users register with CA, and get their global public and private key pair, so do all authorities.

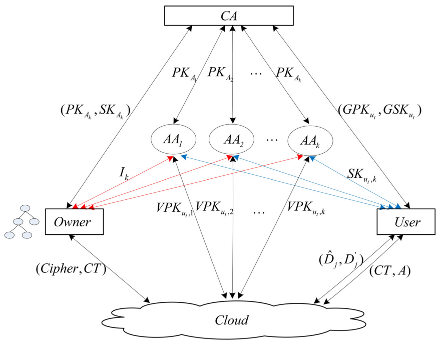


Fig. 1. System Model

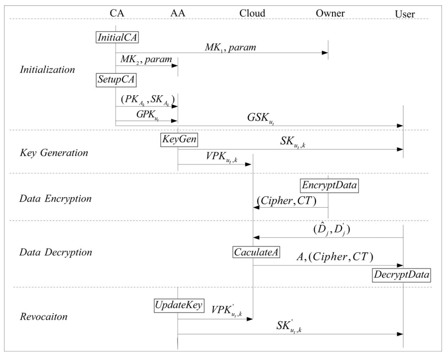


Fig. 2. Framework

AAs. They are key authorities in charge of issuing, revoking, and updating attributes for users according to their roles or rights.

Cloud. It accepts and stores ciphertexts sent from data owners, and provides accesses for users. To decrease the users' decryption cost, we turn over most of decryption operations to cloud storage servers. What is more, the cloud storage servers are responsible for updating version key when their attached attributes are revoked.

Owners. They carry out AONT transform on the original data to *cipher*. From generation matrix G created by AONT, some elements are extracted randomly per line to form a new vector $\bar{A}$ which is to be encrypted by CP-ABE to CT. Then the *cipher* and CT are transmitted to a cloud storage server.

Users. When users want to decrypt ciphertexts after retrieving from cloud, they submit the mapping information of their private keys to cloud. Then, the cloud server generates the mapping ciphertexts A, and sends it back to users. Users recover the plaintexts according to A and their private keys.

3.4 Framework

As illustrated in Fig. 2, our access scheme consists of five parts: ***Initialization***, ***Key Generation***, ***Data Encryption***, ***Data Decryption***, and ***Revocation***.

The algorithm *InitialCA* is run by CA. It generates global parameter *param* and master key *MK*.

The algorithm *SetupCA* is run by CA. It generates a public key and private key pair (PK_{A_k}, SK_{A_k}) for valid authorities. Likewise, it generates a public key and private key pair (GPK_{u_t}, GSK_{u_t}) for valid users.

The algorithm *KeyGen* is run by AAs. It generates private key $SK_{u_t,k}$, version public key $VPK_{u_t,k}$ and version private key $VSK_{u_t,k}$ for each user.

The algorithm *EncryptData* is run by owners. It transforms the original data to a piece of *cipher*, and forms a vector by extracting randomly elements from generation matrix G per line. Then, the vector is to be encrypted by *Tree* with CP-ABE to *CT*. Then, owners send the *cipher* and *CT* to cloud.

The algorithm *CalculateA* is run by a cloud storage server. It takes as input a user's mapping information of his private key, version public key $VPK_{u_t,k}$ stored in cloud and *CT*, and outputs the mapping ciphertexts A.

The algorithm *DecryptData* is run by user. It calls the algorithm *CalculateA*, and computes the plaintext data *M*.

The algorithm *UpdateKey* is run by an authority. It computes new keys for the user. Then, the new private key is issued to the user. Accordingly, the version public and private key pair is recomputed, and the new version private key is sent to the cloud storage server for updating.

3.5 Construction

Let G_0 and G_1 denote bilinear groups with the same prime order p. Let g be generator of group G_0. Let $e : G_0 \times G_0 \rightarrow G_1$ be the bilinear map. Let $H : \{0,1\}^* \rightarrow G_0$ be a hash function.

A. Initialization

Algorithm *InitialCA* gives the detailed description.

Algorithm *InitialCA*

```
input: $\lambda$;
output: $param$, $MK$;
1: CA chooses $G_0$, $G_1$ and $H$ as global parameters $param$;
2: CA chooses random numbers $a, b \in Z_P^*$;
3: $MK_1 \leftarrow e(g,g)^a$;
4: $MK_2 \leftarrow g^b$;
5: $MK \leftarrow (MK_1, MK_2)$;
6: CA sends $MK_1$ and $param$ to owner;
7: CA sends $MK_2$ and $param$ to AA;
```

Let S_A and S_U denote the set of authorities and the set of users. Let $|S_A|$ and $|S_U|$ denote the number of authorities and the number of users. Algorithm *SetupCA* gives the detailed description.

B. Key Generation

Let $I_{u_t,k}$ denote the attributes set of user u_t issued by authority AA_k. Algorithm *KeyGen* gives the detailed description.

C. Data Encryption

Let G and M denote generation matrix and plaintext. Let $M = (m_0, m_2, ..., m_{n-1})^T$. Let $Cipher = (c_0, c_2, ..., c_{n-1})^T$. Let S_T be a set of all attribute leaves of an access tree. Algorithm *EncryptData* gives the detailed description.

D. Cloud-aided Data Decryption

Algorithm *SetupCA*

```
input: $param$;
output: $(PK_{A_k}, SK_{A_k})$, $(GPK_{u_t}, GSK_{u_t})$;
 1: for k =1 to $|S_A|$ do
 2:     CA chooses a random number $a_k \in Z_p$;
 3:     $PK_{A_k} \leftarrow g^{a-a_k}$;
 4:     $SK_{A_k} \leftarrow g^{a_k-b}$;
 5:     CA sends $SK_{A_k}$ to $AA_k$ by the key exchange protocol;
 6: end for
 7: for t =1 to $|S_U|$ do
 8:     CA chooses random numbers $r_t, r_u \in Z_p^*$;
 9:     $GPK_{u_t} \leftarrow g^{r_t}$;
10:     $GSK_{u_t} \leftarrow g^{r_u-r_t}$;
11:     CA publishes $GPK_{u_t}$ within the set $S_A$;
12:     CA issues $GSK_{u_t}$ to user $u_t$ by the key exchange protocol;
13: end for
```

Algorithm *KeyGen*

```
input: $MK_2$, $param$, $PK_{A_k}$, $SK_{A_k}$;
output: $SK_{u_t,k}$, $VPK_{u_t,k}$, $VSK_{u_t,k}$;
 1: AA chooses a random number $v_t \in Z_p^*$;
 2: $VPK_{u_t,k} \leftarrow g^{-v_t}$;
 3: $VSK_{u_t,k} \leftarrow g^{v_t}$;
 4: AA sends $VPK_{u_t,k}$ to cloud by the key exchange protocol;
 5: AA chooses random numbers $r_1, r_2, ..., r_j, ... \in Z_p^*$ for each attribute of user $u_t$;
 6: $D \leftarrow MK_2 \cdot PK_{A_k} \cdot SK_{A_k} \cdot GPK_{u_t}$;
 7: for $\lambda_j \in I_{u_t,k}$ do
 8:     $D_j \leftarrow g^{r_t} \cdot H(\lambda_j)^{r_j}$;
 9:     $D_j^{'} \leftarrow g^{r_j} \cdot VSK_{u_t,k}$;
10: end for
11: $SK_{u_t,k} \leftarrow (D, \{(D_j, D_j^{'})\}_{\lambda_j \in I_{u_t,k}})$;
12: $AA_k$ sends $SK_{u_t,k}$ to user $u_t$, and $VSK_{u_t,k}$ is stored locally;
```

Algorithm *CalculateA* gives the detailed description.

Let S be the set of attributes in the user's private key. F_z denotes the result of $DecryptNode_x$ run on his child z of node x. I_x denotes the set of children of node x. The algorithm $DecryptNode_x$ is called from the root node R of the access tree in a recursive way. Algorithm $DecryptNode_x$ gives the detailed description.

After retrieving A and $(Cipher, CT)$ from cloud, user can call the algorithm *DecryptData* to compute the plaintext M. Algorithm *DecryptData* gives the detailed description.

The cloud cannot get D_j from $\hat{D}_j$ because user's global private key is unknown. Meanwhile, the vector $\bar{A}$ can only be got by matching $SK_{u_t,k}$ to $VSK_{u_t,k}$ and access tree. Generation matrix G can be reconstructed only when the vector $\bar{A}$ is achieved.

Algorithm *EncryptData*

```
input: $MK_1$, $param$, $G$, $M$;
output: ($Cipher$, $CT$);
 1: Owner splits original data to fragments $M$;
 2: $Cipher \leftarrow G \times M$; //Owner transforms M to Cipher. To keep data confidentiality,
    the generation matrix $G$ used in AONT must be of full rank, and be randomly
    generated for each run of AONT.
 3: for i=0 to $l-1$ do
 4:     Owner chooses a random number $r_i \in Z_n$ and a random element $g_{i,r_i} \in G$;
 5:     Owner appends $g_{i,r_i}$ to $\bar{A}$;
 6: end for
 7: $\hat{C} \leftarrow \bar{A} \cdot e(g,g)^{a \cdot s}$;
 8: $C \leftarrow g^s$;
 9: for $\lambda_i \in S_T$ do
10:     $C_i \leftarrow g^{q_i(0)}$;
11:     $C_i^{'} \leftarrow H(\lambda_i)^{q_i(0)}$;
12: end for
13: $CT \leftarrow (T, \hat{C}, C, \{(C_i, C_i^{'})\}_{\lambda_j \in S_T})$;
14: Owner sends $(Cipher, CT)$ to cloud;
```

Algorithm *CalculateA*

```
input: $\left\{\hat{D}_j, D_j^{'}\right\}_{\lambda_j \in I_{u_t,k}}$, $VPK_{u_t,k}$, $CT$;
output: $A$;
 1: for $\lambda_j \in I_{u_t,k}$ do
 2:     $\hat{D}_j^{'} \leftarrow D_j^{'} \cdot VPK_{u_t,k}$;
 3: end for
 4: $A \leftarrow DecryptNode_R$ ;
 5: Cloud sends $A$ and $(Cipher, CT)$ to user $u_t$;
```

E. Attribute Revocation
When the attribute λ_{x,u_t} of user u_t is revoked, AA chooses random numbers $v_t^{'} \in Z_p^*, v_t^{'} \neq v_t$, and updates the user's private key $SK_{u_t,k}^{'}$. Meanwhile, the new version key pair $(VPK_{u_t,k}^{'}, VSK_{u_t,k}^{'})$ is updated. Algorithm *UpdateKey* gives the detailed description.

4 Analysis

4.1 Security

A. Collusion Resistance
We consider that the private key of user u_p is $SK_{u_t^p,k}$. Assume that the user u_q want to combine his private key with $SK_{u_t^p,k}$. As a result, the user u_p achieves

Algorithm *DecryptNode*$_x$

```
input: $S$, $CT$, $\left\{\hat{D}_j, D_j^{'}\right\}_{\lambda_j \in I_{u_t,k}}$;
output: $DecryptNode_x$;
 1: if the node $x$ is a leaf of the access tree $T$ then
 2:     Let $\lambda_i$ be the attribute of the node $x$;
 3: end if
 4: if $\lambda_i \in S$ then
 5:     $DecryptNode_x \leftarrow \frac{e(\hat{D}_i, C_i)}{e(\hat{D}_i^{'}, C_i^{'})}$;
 6: end if
 7: if $\lambda_i \notin S$ then
 8:     $DecryptNode_x \leftarrow \perp$;
 9: end if
10: if $x$ is non-leaf then
11:     $i \leftarrow index(z)$;
12:     $s_x \leftarrow \{index(z), z \in I_x\}$;
13:     $F_x \leftarrow \prod_{z \in I_x} F_z^{\Delta_{i,s_x}(0)}$;
14: end if
15: if $x$ is the root of the access tree then
16:     Cloud outputs $DecryptNode_x$;
17: end if
```

Algorithm *DecryptData*

```
input: $SK_{u_t,k}$, $Cipher$, $CT$, $A$;
output: $M$;
 1: $\hat{D}_j \leftarrow D_j \cdot GSK_{u_t}$; //The user $u_t$ attaches their global private key $GSK_{u_t}$to the
    part of attribute key;
 2: The user $u_t$ sends $(\hat{D}_j, D_j^{'})$ to cloud, and gets $(Cipher, CT)$ and $A$;
 3: $\hat{D} \leftarrow D \cdot GSK_{u_t}$;
 4: $\bar{A} \leftarrow \frac{\hat{C} \cdot A}{e(C, \hat{D})}$;
 5: The user $u_t$ reconstructs the generation matrix $G$;
 6: The user $u_t$ computes the inverse matrix $G^{-1}$;
 7: The user $u_t$ computes the plaintext data by $Cipher$ and $G^{-1}$;
```

his new private key:

$$\begin{aligned} SK_{u_t^p,k} =& (D, \{(D_j, D_j^{'})\}_{\lambda_j \in I_{u_t^p,k}} \cup \{(D_j, D_j^{'})\}_{\lambda_j \in I_{u_t^q,k}}) \\ =& (g^a \cdot g^{r_t^p}, \{(g^{r_t^p} \cdot H(\lambda_j)^{r_j}, g^{r_t^p} \cdot g^{v_t^p})\}_{\lambda_j \in I_{u_t^p,k}} \\ & \cup \{(g^{r_t^q} \cdot H(\lambda_j)^{r_j}, g^{r_t^q} \cdot g^{v_t^q})\}_{\lambda_j \in I_{u_t^q,k}}), \end{aligned}$$

where r_t^p and r_t^q are random numbers chosen by CA for user u_p and u_q respectively. v_t^p and v_t^q are random numbers chosen by AA for different users respectively.

```
Algorithm UpdateKey
input: $SK_{u_t,k}$, $VPK_{u_t,k}$, $VSK_{u_t,k}$;
output: $SK'_{u_t,k}$, $VPK'_{u_t,k}$, $VSK'_{u_t,k}$;
 1: AA chooses a random number $v'_t \in Z^*_P$;
 2: $VPK'_{u_t,k} \leftarrow g^{-(v_t - v'_t)}$; //AA computes user's new version key pair.
 3: $VSK'_{u_t,k} \leftarrow g^{v_t - v'_t}$;
 4: AA sends $VPK'_{u_t,k}$ to user cloud;
 5: $D \leftarrow g^a \cdot g^{r_t}$;
 6: for $\lambda_j \in I_{u_t,k} \backslash \lambda_{x,u_t}$ do
 7:     $D'_j \leftarrow g^{r_j} \cdot g^{v_t - v'_t}$;
 8:     $D_j \leftarrow g^{r_t} \cdot H(\lambda_j)^{r_j}$;
 9: end for
10: $SK'_{u_t,k} \leftarrow (D, \{(D_j, D'_j)\}_{\lambda_j \in I_{u_t,k} \backslash \lambda_{x,u_t}})$; // $SK'_{u_t,k}$ is user's new private key.
11: AA sends $SK'_{u_t,k}$ to user $u_t$;
```

To decrypt a CT, the algorithm *CalculateA* is run as follows:

Step1: $\forall \lambda_j \in I_{u^q_t,k}$, $\hat{D}'_j = D'_j \cdot VPK_{u^p_t,k} = g^{r_j} \cdot g^{v^q_t} \cdot g^{-v^p_t}$.

Step2: $\forall \lambda_j \in I_{u^q_t,k}$, $DecryptNode_x = \frac{e(\hat{D}_i, C_i)}{e(\hat{D}'_i, C'_i)} = \frac{e(g^{r^p_u}, g^{q_i(0)})}{e(g^{v^q_t - v^p_t}, H(\lambda_i)^{q_i(0)})}$.

Because $v^p_t \neq v^q_t$, the parameter s cannot be recovered by Lagrange's interpolation method when $DecryptNode_x$ is run in a recursive way. That is, even if the user colludes with another, he cannot get more rights.

B. Data Confidentiality
The plaintext data is transformed to *Cipher*, which is stored with CT in cloud storage server. Owners encrypt the part of generation matrix, namely the vector $\bar{A}$. Only the user whose attributes meet access tree can decrypt CT to the vector $\bar{A}$. With the vector $\bar{A}$, the user can recover generation matrix G, and work out the inverse matrix G^{-1} to carry out the final calculation of the plaintext $M = Cipher \cdot G^{-1}$. Unauthenticated users and cloud storage server cannot recover the plaintext M even if they have got the tuple $(Cipher, CT)$.

C. On-demand Revocation
Our scheme enforces the attribute revocation immediately that can achieve both *forward security* and *backward security.* When a user joins, CA issues his global public key and private key pair (GPK_{u_t}, GSK_{u_t}). After that, the AA distributes his private key $SK_{u_t,k}$ and version key pair ($VPK_{u_t,k}$, $VSK_{u_t,k}$). Because the new user has a private key matching version key stored in cloud, he can decrypt the previously published ciphertexts, only if his attributes meet access tree. Meanwhile, our scheme can assure *backward security.* When a user's attribute is revoked, he fails to decrypt new ciphertexts with previous $SK_{u_t,k}$, because the version key is mismatched.

4.2 Efficiency

A. Comprehensive Analysis
Table 1 shows the comparison between our scheme and two existing schemes. Our scheme spends less computation cost than Hur's on decryption on the user. Compared with DAC-MACS, our scheme spends less computation cost on encryption on the owner, and less cost on cloud-aided decryption. Most notably, our scheme can revoke user's attribute without updating other users' keys or re-encrypting ciphertexts.

Table 1. Comparison of computation cost

		Hur's scheme[12]	DAC-MACS[3]	Our scheme
Owner(Encryption)		(2t + 2)exp	(6t + 3) exp	(2t + 2)exp
User(Decryption)		(k + log$t$) exp+ (2k + 1) pairing	exp	exp + pairing
Cloud(Decryption)		0	$N_A \cdot ((\sum_{k=1}^{I_{A_k}}$ (3pairing + exp))+ 2pairing)	(k + log t) exp+ (2k) pairing
Revocation	Revocation applicants	User	AA	AA
	Key updating of other users	Yes	Yes	No
	Ciphertext re-encryption	Yes	Yes	No

exp: exponent calculation. pairing: $e(g, g)$. t: the number of attributes in access tree. k: the number of attributes in user's private key. I_{A_k}: the set of attributes issued by AA_k in ciphertext. N_A: the number of AA_k in ciphertext.

B. Storage Overhead
Table 2 summarizes the comparison results of storage overhead between previous schemes and our scheme. In the comparison, our scheme achieves the lowest storage overhead on each owner than previous schemes. We achieve the highest efficiency of attribute revocation through sacrificing storage overhead on each user and AA_k.

In our scheme, the storage overhead on AA_k consists of master key MK_2, the public key PK_{A_k} of AA_k, the private key SK_{A_k} of AA_k, the set of attributes managed by AA_k, and the public key GPK_{u_t} of users u_t. The storage overhead on each owner comes from attributes issued by AA_k and master key MK_1. The storage overhead on each user is composed of user's global public key and private key pair (GPK_{u_t},GSK_{u_t}) and attribute private key $SK_{u_t,k}$. The tuple $(Cipher, CT)$ and version key $VPK_{u_t,k}$ make the main storage overhead on the cloud storage server.

5 Simulation

We simulate the computing time of revocation, which shows that our scheme incurs less computation cost when the attribute of a user is revoked.

Table 2. Comparison of storage overhead ($|p|$)

	Owner	User	AA_k	Cloud
Hur's scheme[12]	$2N_A + \sum_{k=1}^{N_A} n_{a,k}$	$1 + 2\sum_{k=1}^{N_A} n_{a,k,u_t}$	$n_{a,k} + 3$	$2 + 2t$
DAC-MACS[3]	$3N_A + 1 + \sum_{k=1}^{N_A} n_{a,k}$	$3N_A + 1 + \sum_{k=1}^{N_A} n_{a,k,u_t}$	$n_{a,k} + 3$	$3 + 3t$
Our scheme	$1 + \sum_{k=1}^{N_A} n_{a,k}$	$3 + 2\sum_{k=1}^{N_A} n_{a,k,u_t}$	$n_{a,k} + 3 + n_{u_t,k}$	$3 + 2t$

$n_{a,k}$: the number of attributes managed by AA_k. n_{a,k,u_t}: the number of attributes issued by AA_k to user u_t. $|p|$: the storage overhead of a element in group G. $n_{u_t,k}$: the number of users managed by AA_k .

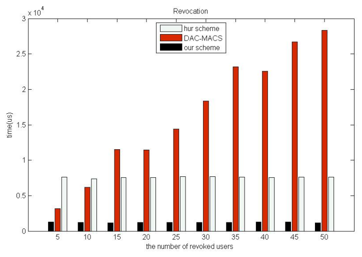
The simulation is finished on an OpenSUSE system with an Intel Core 2 CPU at 2.8GHz and 1GB RAM. The Pairing-Based Cryptography library (PBC) is used to simulate in coding. α is chosen as elliptic curve, 160 bit as group order, and 512bit as field size. Let *n_u* denote the number of involved users. Let *n_c* denote the number of ciphertexts encrypted with the revoked attribute. Let *n_attr_c* denote the number of attributes in each ciphertext. Let *n_attr_k* denote the number of attributes in the revoked user private key. Times in Fig.3 are the mean of 100 trials to avoid the results of accidents.

We compare the computation efficiency of revocation. Assume that the revocation time consists of three parts: computational time on AA, computational time on each user, and computational time on each ciphertext. Fig.3(a) describes the comparison of revocation time versus the number of involved users, where *n_c*, *n_attr_c* and *n_attr_k* are all set as 20. Fig.3(b) gives the comparison of revocation time versus the number of ciphertexts encrypted with the revoked attribute, where *n_u*, *n_attr_c* and *n_attr_k* are all set as 20. Fig.3(c) shows the comparison of revocation time versus the number of attributes in each ciphertext, where *n_u*, *n_c*, and *n_attr_k* are all set as 20. Fig.3(d) describes the comparison of revocation time versus the number of attributes in the revoked user private key, where *n_u*, *n_c*, *n_attr_c* are all set as 20.

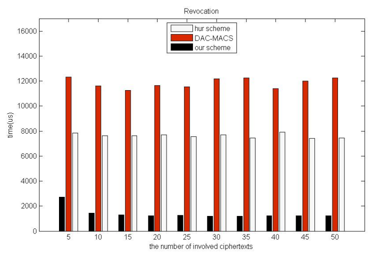
Fig.3(a) shows the revocation time is linear to the number of involved users in DAC-MACS[3]. Fig.3(c) and Fig.3(d) show the revocation time is linear to the number of attributes in each ciphertext or the number of attributes in the revoked user private key respectively in Hur's scheme[12]. Our scheme spends less time because the revocation of user's attribute just needs to update his $SK_{u_t,k}$ and $VPK_{u_t,k}$ without updating other users key or re-encrypting ciphertexts.

6 Conclusions

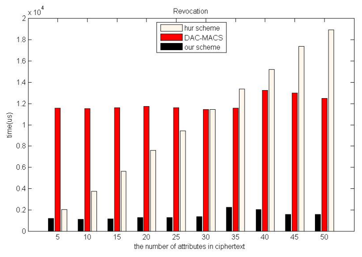
We construct an access control scheme based on CP-ABE in cloud storage. In this work, the computational cost of encryption is lower, and the main computation of decryption is transferred to the cloud server. We also design an attribute



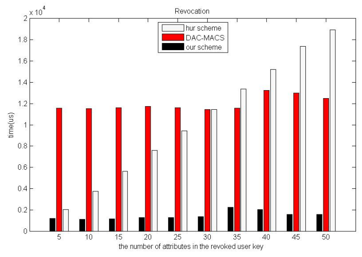
(a) n_c=20; n_attr_c=20; n_attr_k=20



(b) n_u=20; n_attr_c=20; n_attr_k=20



(c) n_u=20; n_c=20; n_attr_k=20



(d) n_u=20; n_c=20; n_attr_c=20

Fig. 3. The comparison of revocation time

revoking method with version key so that both *forward security* and *backward security* are achieved. Meanwhile, the scheme can resist collusion of users and the problems due to key escrow. However, the scheme does not consider the collusion between AAs and users. That is to say, a user can get permission if he colludes with AAs.

Acknowledgments. This work is supported by the National Science Foundation of China (No. 61373040, No.61173137), The Ph.D. Programs Foundation of Ministry of Education of China (20120141110073), Key Project of Natural Science Foundation of Hubei Province (No. 2010CDA004).

References

1. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
2. Hohenberger, S., Waters, B.: Attribute-based encryption with fast decryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 162–179. Springer, Heidelberg (2013)

3. Yang, K., Jia, X., Ren, K., et al.: Dac-macs: Effective data access control for multi-authority cloud storage systems. In: 32th IEEE INFOCOM, pp. 2895–2903 (2013)
4. Yang, K., Jia, X.: DAC-MACS: Effective Data Access Control for Multi-Authority Cloud Storage Systems. In: Brauer, W. (ed.) GI 1973. LNCS, vol. 1, pp. 59–83. Springer, Heidelberg (1973)
5. Yang, K., Jia, X.: Attributed-based access control for multi-authority systems in cloud storage. In: 32th IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 536–545 (2012)
6. Bobba, R., Khurana, H., Prabhakaran, M.: Attribute-sets: A practically motivated enhancement to attribute-based encryption. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 587–604. Springer, Heidelberg (2009)
7. Wan, Z., Liu, J., Deng, R.H.: HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing. J. IEEE Transactions on Information Forensics and Security 12(7), 743–754 (2012)
8. Wan, Z., Liu, J., Zhang, R., et al.: A Collusion-Resistant Conditional Access System for Flexible-Pay-Per-Channel Pay-TV Broadcasting. J. IEEE Transactions on Multimedia 15(6), 1353–1364 (2013)
9. Ruj, S., Nayak, A., Stojmenovic, I.: DACC: Distributed access control in clouds. In: 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 91–98 (2011)
10. Zhu, Y., Hu, H., Ahn, G.J., et al.: Towards temporal access control in cloud computing. In: 31th IEEE INFOCOM, pp. 2576–2580 (2012)
11. Li, J., Huang, Q., Chen, X., et al.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: 6th ACM Symposium on Information, Computer and Communications Security, pp. 386–390 (2011)
12. Hur, J., Kang, K.: Secure Data Retrieval for Decentralized Disruption-Tolerant Military Networks. J. IEEE/ACM Transactions on Networking 22(1), 16–26 (2014)
13. Li, M., Yu, S., Zheng, Y., et al.: Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. J. IEEE Transactions on Parallel and Distributed Systems. 24(1), 131–143 (2013)
14. Wu, Y., Wei, Z., Deng, R.H.: Attribute-Based Access to Scalable Media in Cloud-Assisted Content Sharing Networks. J. IEEE Transactions on Multimedia. 15(4), 778–788 (2013)
15. Müller, S., Katzenbeisser, S.: Hiding the policy in cryptographic access control. In: Meadows, C., Fernandez-Gago, C. (eds.) STM 2011. LNCS, vol. 7170, pp. 90–105. Springer, Heidelberg (2012)
16. Attrapadung, N., Imai, H.: Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 278–300. Springer, Heidelberg (2009)

Full and Live Virtual Machine Migration over XIA[⋆]

Dalu Zhang[⋆⋆], Xiang Jin, Dejiang Zhou, Jianpeng Wang, and Jiaqi Zhu

Department of Computer Science and Technology, Tongji University, Shanghai, China
daluz@acm.org,
{jinxiang8910,dejiang_zhou,wangjianpeng4321,garyzjq}@163.com

Abstract. Future network research for replacing the current network is becoming more and more imperative, since traditional TCP/IP network is showing numbers of shortcomings such as address depletion, scalability and security issues. Meanwhile, VM migration technique is crucial in cloud computing. FIA (Future Internet Architecture) projects are supported by US NSF in 2010 for future Internet design. Among the projects, XIA is the one which complies with clean slate concept thoroughly, discarding TCP/IP protocol stack. Unfortunately, its discussion on VM migration technology is imperfect. This paper is an experimental study aims at verifying the feasibility of VM migration over XIA. The procedure is managed by a migration control protocol which meets the characters of XIA. An elementary self-adaptive mechanism is introduced to maintain connectivity of VM and service connection states, which is also beneficial for VM migration in TCP/IP network. Evaluation results show that our solution can support live VM migration in XIA effectively with services uninterrupted.

Keywords: XIA, VM migration, chunk, self-adaptive, downtime.

1 Introduction

Internet is indubitably one of the most useful tools in our daily life. Developed based on TCP/IP protocol, Internet has achieved great flourish not only because of large quantity of applications, but also various kinds of media accessing to it. Network at present is exposing more and more serious issues such as the shortage of addresses, difficulties on scalability, mobile device mobility and security threats. This leads to the development of future network design.

US NSF supported five FIA (Future Internet Architecture) projects in 2010. Among them, XIA [1] is the one which aims at getting rid of TCP/IP concepts. Thus, it does not suffer from the shortage of IP addresses. Network, host, service and content can be abstracted as principals in XIA. New principal types can be defined for special use, even if the network can't natively support the new

[⋆] Supported by the NSF of China (No. 61073154).
[⋆⋆] Corresponding author.

function [2]. Network address is expressed by DAG (Directed Acyclic Graph), which is flexible for addressing. Each node in the DAG represents a principal type respectively. In addition, *fallback* path allows communicating entities to choose an alternative action if intent node is unreachable. Name service runs on a name server host for address management, which providing a mapping from human-readable names to DAGs.

Virtualization is necessarily one of the key technologies in cloud computing, which allows to run multiple operating systems on a single platform. This can help to raise the working efficiency of expensive computing resources on a physical host, especially CPU cycles and memory space. Data centers can achieve load balancing, host maintenance, energy management, network resilience [3] or disaster recovery [4] by migrating VMs.

Extensive research has been carried out on shared-storage VM migration [5], such as NFS (Network File System). But the fact is that shared-storage VMs cannot be applied in all scenarios. For example, a user may not necessarily access to a specific data center permanently [6]. If a shared disk is configured for his VM through network, QoS would be affected greatly by abnormal latency. Therefore, we propose full migration, during which virtual disk is transferred as well as CPU and memory content. Fortunately, it is well supported in QEMU-KVM.

KVM (Kernel-based Virtual Machine) is a piece of software frequently used in research fields. KVM module is often integrated in the kernels of common Linux distributions. Kernel modules in Linux OS cannot be controlled directly users and this is the task of QEMU. QEMU is also a piece of open-source virtualization software and is adopted as a management tool in user space.

Total migration time and downtime [7] are two of the main parameters that should be taken into consideration for VM migration. Total migration time refers to the total time needed to move a VM from one host to another. Downtime indicates the period of time when VM is not running and application degradation is perceived by user. In future Internet, total migration time and downtime would be affected by the new characters, such as new methods for addressing and data transmission.

In this paper, we present VM migration primarily based on XIA network. We firstly design VM migration platform in two different environments of XIA, namely in a single AD (Administration Domain) and between ADs. Since VM migration over XIA network is quite different to that over TCP/IP network, we adopt a VM migration control protocol to manage migration procedure. Moreover, a current-network based self-adaptive mechanism is introduced to keep all the hosts and VMs communicating normally. Functions of this mechanism include perceiving VM's mobility, propagating new location address messages and recovering all the traffic of VM. We achieve to conduct VM migration over XIA with the control protocol and self-adaptive mechanism presented.

The rest of this paper is organized as follows: In Section 2 and 3, we discuss the motivation for this research and the related works. We demonstrate migration control protocol and self-adaptive mechanism in Section 4. After that, we introduce the VM migration system design and processing modules in Section 5. Section 6 further discusses the experiment results and evaluates migration performance. Section 7 concludes this paper and list some special issues leave for future research.

2 Motivation

Virtualization technology is generated to make a full use of computing resources. Thus, it is still necessary for maintaining unified management of various cloud computing platforms, even in future data centers. In addition, virtualization is also beneficial to keep user diversity and application isolation. Therefore, virtual machine and its migration technique will be in existence for a long period of time in the future.

As a typical application in future networks, VM and migration technique also should be well supported in XIA. There are many research about VM migration over traditional TCP/IP network, but few of them are related to future Internets. Therefore, we try to conduct VM migration over future Internet project XIA. Firstly, it can test whether VM migration can be well supported in XIA network. Secondly, in contrast to the technologies used in TCP/IP network, we can find out what is needed in XIA to achieve the goal. The solutions can also be introduced into VM migration over TCP/IP, getting rid of the mechanisms such as tunnels or agents that are often used. Our research results may be a reference for XIA research community and a contribution to future network design.

3 Related Work

Most recent research on VM migration is dedicated in studying the principles of VM migration. In general, VM migration method can be classified as pre-copy [8], post-copy [9]. There are also optimizations based on pre-copy algorithms, such as transferring bitmap [10] or log file [11] of dirty pages, delaying hot pages' delivery to the last round [12] to minimize the number of pages that are being transferred.

For VM migration prediction and strategy, migration can be triggered by the access time of network file [13] or determining whether the total transmission time exceeds a specific threshold [14]. [15] proposes to choose a server that needs to be migrated in order to reduce downtime and energy consumption. Other parameters like CPU, memory size, bandwidth [16] and throughput [17] are also introduced in migration prediction and decision.

Lots of research has been carried out in TCP/IP network to solve the problem of service interrupt when VM migration occurs in WAN. They can be classified into two categories. One is based on the concept of mobile IP. [18] presents to build a tunnel between former address and the new address in order to keep all the communications that have been established before. Besides, dynamic DNS is utilized to record address update and provide new address to clients. Network agents [19] are presented to be set in both source and destination subnet with ARP agents maintained on them. ARP agent in source subnet broadcast unsolicited ARP messages to advertise its new location. But the limitation is that application users should also be located in source subnet, otherwise, they cannot receive ARP messages. Mobile IPv6 is adopted in [20], one obvious benefit is that hosts supporting Mobile IPv6 can bypass the tunnel and connect to VM through route optimization mode.

The other is a method based on overlay network. In [21], the source and destination network of VM migration procedure are repartitioned into the same VPN. ARP message can be forwarded at VPN level to update Ethernet switch's mappings at both sites. This will help to redirect network connections to the VM's new location. In ViNe [22], hosts can be addressed by virtual network addresses first. Overlay network method requires virtual network establishment before migration occurs. Virtually, VM migration event is triggered by particular factors, virtual networks have to be reorganized each time since the source and destination network is not fixed.

Seldom research has been carried out talking about VM migration technologies over future Internet architectures. Therefore, it is contributory for us to present VM migration research on XIA. It is actually contributive for future network design.

4 Control Protocol and Self-adaptive Mechanism

Two basic issues should be taken into consideration when conducting VM migration in a new network. One is how to deliver the migration data and the other is to keep VM's service connections. On one hand, CHUNK is introduced to be a way for data transmission in XIA. Data sender would not deliver chunks directly to receiver in a session. It first tells receiver the CIDs of chunks and then the receiver starts to request for data according to the CIDs. We present a VM migration control protocol to manage this procedure.

On the other hand, in implementation of VM migration in WAN, the VM will get unreachable after being migrated to destination subnet. There are mainly three challenges. Firstly, the IP address obtained before migration belongs to the source subnet and can't be recognized in new subnet, even the destination host which the VM lies on can't be aware of VM's existence. Secondly, even if a new address is acquired, it is hard to get the new address propagated to the whole Internet. Establishing the relationship between the obsolete address and the new one is also difficult. Thirdly, communications related to the migrated VM should be recovered. Additionally, connections that are set up afterward must be routed to the right location. In our pre-experiments, we find that similar issues exist in VM migration in XIA network. Analogically, we introduce self-adaptive mechanism to solve the problem.

4.1 VM Migration Control Protocol

The control messages are delivered by STREAM because it is simple and reliable. Fig. 1 shows the process of data transmission. The destination node first start a stream socket and bind it to a migration service. It runs in listening state,

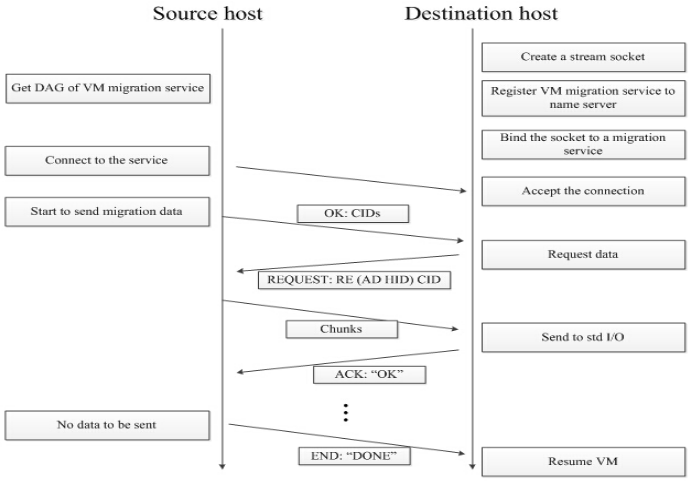


Fig. 1. Detailed working process of VM migration control protocol

waiting for connection of migration data sender host. After connection is set up, the source host (sender) will notify the destination host (receiver) of chunks CIDs. The destination host will construct messages to request for these chunks. The receiver then acknowledges for this round of transmission if the data are check out to be correct. The procedure is repeated till nothing to be delivered. When it comes to the end of migration, a "DONE" message will be sent out to announce the termination of VM migration.

4.2 Self-adaptive Mechanism

We present self-adaptive mechanism after VM is resumed on destination host. In order to solve the problems above, self-adapt procedure mainly focuses on three aspects, namely, mobility perception, new location notification and traffic redirection.

Mobility Perception: One basic precondition to complete VM migration procedure and recover all the traffic is that the mobility of VM should be detected in time. A general solution is to intercept and capture signals from the hypervisor when particular event occurs. A simple alternative strategy can be to make aware of mobility by notifications from migration sending or receiving processes. This goal can be achieved conveniently in XIA. The router inside an AD broadcast messages periodically which contain identifiers of the AD and router, and the location of name service. Any hosts that receive these broadcast packets can easily determine which administrative domain they belong to at present.

New Location Notification: Address in XIA is expressed by DAG and a common form is depicted in Fig. 2. This structure is constructed with nodes

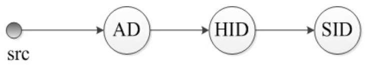


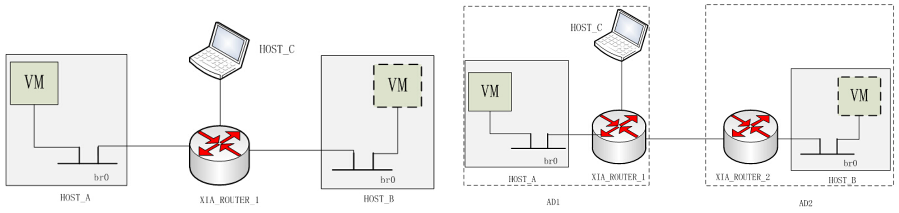
Fig. 2. An example of DAG form address (*src* indicates a virtual source of DAG. *AD* is a 160-bits number identifying an administrative domain. Similarly, *HID* and *SID* are 160-bits identifiers presenting a host or a service.).

representing different principal types, so it is convenient in re-construction. The AD number will be changed when a virtual machine is migrated to a new AD, and a new DAG form address should be re-registered to name service as soon as mobility is detected. Additionally, since a router is to manage the AD it locates in, the new migrating virtual machine should also make itself noticed by the gateway router in the destination AD and the router will append its routing table with an entry directing to the VM.

Traffic Redirection: Most of the research that studying live VM migration in WAN adopt agents on source host and tunnels between agent and VM. Neither agents nor tunnels are needed when it comes to XIA because of its particular characteristics. IP address is no more used for addressing in XIA. Bytes filled in either source or destination address field of XIP layer header is DAG instead. *Lastnode* field in XIP header stores an identifier that indicates the node in DAG which is last processed. When a router receives a packet, it checks the *lastnode* field first and then processes the nodes afterward. Routing is determined according to the next node in the DAG.

In order to keep up communications after migration, we can take some modifications to the routing tables of routers on the migration path. VM's DAG is changed after VM is migrated to another AD, but application clients still use the original DAG for transmission. All the packets to VM will first go to source AD. Thus, we just have to change the *next-hop* field of VM's HID entry to make the path direct to new location of VM. If a router in source AD receives a packet whose destination address is VM's obsolete DAG, it will direct that packet to the next router which is nearer to destination host. Though the information of AD is invalid when a VM is migrated, packets directing to the VM are still able to be delivered correctly according to HID routing entries.

Besides, when a VM is running on source host, a HID entry is added into it with host's HID as destination and next-hop address. Similarly, there is also an entry directing to HID_VM in source host's routing table. This will lead to trouble as both of the entries can't get modified automatically and packets will be routed incorrectly. Therefore, these two entries should be deleted in order to keep connectivity between source host and migrated VM, both in intra-AD and inter-AD migration.



(a) Intra-AD **(b)** Inter-AD

Fig. 3. VM migration testbed design in XIA network, (a) is the testbed in single AD and (b) is a typical testbed for VM migration between different ADs

5 System Establishment

5.1 Testbed Design

The XIA prototype runs on top of the software router *click* [23] designed in MIT. Common routers in TCP/IP network can't be recognized by XIA prototype because of the different protocol formats. As a solution, there are at least two modes that can be chosen from by each machine, that is, to be a host or a router. Multiple network interface cards are needed when a physical machine runs as a router and then it can process and route packets that supporting XIA protocol.

Name service is necessary for hosts' dynamic deployment. Any host can run as name server and provide global name resolution service. When the addresses of hosts or services are changed, they will be registered to name server.

Our goal of the research can be defined as four rules which is named as *"four any"*, that is, virtual machines can be deployed on any physical hosts in the XIA network, VM can be migrated to any host, name service can run on any host in the network and any of the applications should not be interrupted during VM migration. It means that any of the virtual machines in the environment can be migrated at any time, assuming that security is guaranteed.

AD is introduced in XIA for network management. A VM in different AD will obtain different addresses since the AD number is changed. Therefore, inter-AD VM migration is more complex than intra-AD VM migration in XIA, just as that in TCP/IP network. We propose two VM migration testbeds, in single AD and between ADs, concerning the issue of whether DAG has to be changed.

Fig. 3a shows the testbed of VM migration in single AD. HOST_A, HOST_B and HOST_C represent the source host, destination host and client host respectively. HOST_VM depicted in dashed box denotes the virtual machine running on source host before migration. Besides, name service runs on HOST_A because it can be put on any of the hosts. ROUTER_1 indicates the XIA router in this AD, routing and forwarding packets. In Fig. 3b, the topological structure is partitioned into two independent ADs and each XIA router manages its AD respectively. Name service still runs on HOST_A. All the hosts or services can register their DAG-style addresses to name service.

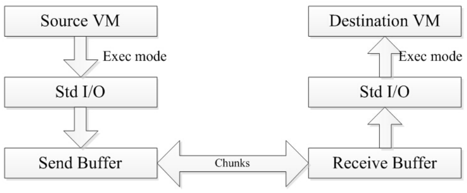


Fig. 4. Data flow between different modules for exec mode VM migration

5.2 Migration Control Modules

Our virtual machine migration platform in XIA network is set up based on the now available virtualization product, so the techniques can be versatile for other network types. We choose KVM as the hypervisor mainly because it can support full migration (storage migration) effectively. We do not focus on iteration phases during migration or factors that trigger migration. Therefore, we don't have to modify the source code of QEMU-KVM. This is beneficial for the now popular virtualization product to be blossom in the coming future when future networks such as XIA takes the place of TCP/IP. Fig. 4 shows the relationship of migration processing modules.

Migration Data Sending and Receiving: *Tcp* and *exec* are two common ways utilized in KVM for migration. *Tcp* mode is designed primarily for VM migration in TCP/IP network, its application interfaces are well designed and can be used directly. In *exec* mode, migration data are read and sent to standard I/O, while on the receiver side KVM obtain data from the standard I/O and then reload the virtual machine, no matter how data are transferred.

Data Delivery: Three data transmission methods are provided in XIA, they are STREAM, DGRAM and CHUNK. STREAM is connection oriented and provides reliable transmission, just as TCP in current TCP/IP protocol stack. Correspondingly, DGRAM is a connectionless mode like UDP. Idea of CHUNK is widely adopted in content-centric future internet architectures, especially in XIA and NDN. Data transmitted are divided into chunks and chunk is regarded as a transmission unit. CID of a chunk is obtained by hash of the whole content block, so it can get self-verified. Besides, network traffic is well controlled because each transaction is originated by the data receiver; the sender just needs to put the data that are required into content cache.

CHUNK mode is quite reliable because of its error control and traffic control mechanisms though it is connectionless. Considering the advantages of CHUNK in future network, we employ it in VM migration procedures in XIA. Migration sending process acquires chunks from standard I/O and delivers them. Then the migration receiving process accepts the chunks and writes them into standard I/O. The details of chunk mode data transmission procedure and the control protocol for VM migration management has been introduced in Section 4.1.

Migration Test and Verify: We propose to run applications in VM during migration so as to determine whether the migration procedure is live or not. Since the VM and all the hosts are configured with XIA network environment, traditional applications cannot work efficiently. Therefore, a calculation application (denoted as Cal in tables) is introduced during VM migration which is developed by using API functions provided by XIA. A calculation server runs in the migrated VM, calculating and verifying *Goldbach* conjecture (every even number can be expressed as a sum of two prime numbers). A client process runs on the client host, acquiring the results and printing them to screen.

Ping is a common but effective tool used for testing network connectivity in TCP/IP network. It can also be utilized to measure migration downtime by sending ICMP messages to VM periodically. We propose to use *xping* provided in XIA prototype to achieve the function of ping. For example, if the time interval of xping packets is set as Δt and n packets are dropped during the time when VM is shutdown, we can get informed that downtime is $n * \Delta t$ with deviation of Δt, that is, the downtime measured is $(n \pm 1) * \Delta t$. It can be quite accurate if value of Δt is small enough.

6 Implementation and Evaluation

We carry out virtual machine migration over XIA network with the testbeds depicted in Section 5. With the evaluation results, we can easily get to know the deficiencies of our design and improve our work in next phase. Kernel-based Virtual Machine (KVM) is chosen as virtual machine hypervisor, with QEMU as a management tool. Full migration is proposed to the VM is configured with 4GB virtual disk and 640MB physical memory. The hosts are configured with Intel core I3 processor and 8GB RAM. Computers with multi network interface cards are used as XIA routers which are different from traditional routers. All the physical hosts and virtual machines are running Linux systems (Ubuntu 12.04) with kernel version 3.5.0. XIA prototype source code package can be obtained from Github. The latest version so far is v1.1.

6.1 Comparison of Migration Modes

Data are delivered by TCP connections when QEMU-KVM runs *tcp* as default mode for migration. In contrast, the method for data transmission can be self-defined by users in *exec* mode. *Tcp* mode cannot be adopted here since TCP connection is not supported in XIA, even though it costs shorter migration time. We choose to get KVM migration run in exec mode. First of all, it is necessary to compare migration performance in *tcp* and *exec* mode so as to get the discrepancy between them. We conduct exec mode data transmission in TCP/IP network by calling SOCK_STREAM sockets and APIs. Two kinds of workload as follows are introduced in our experiment, which are widely used in network research areas.

Dbench: an open source benchmark tool to generate I/O workloads, simulating a variety of real file servers. We choose it as an I/O intensive application.

Netperf: a benchmark that can be used to measure the performance of many different types of networks. Here we introduce it just to be a workload inside VM. It does not communicate with clients because of limitations of future network. It can also be regarded as an I/O application.

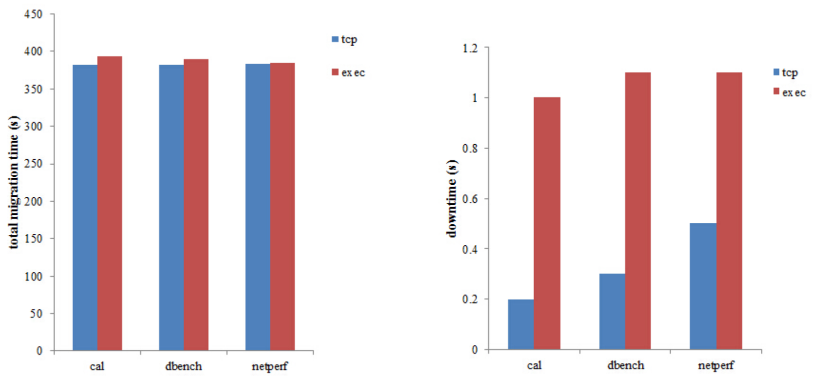


Fig. 5. Total migration time and downtime of two migration mode (*tcp* and *exec*)

Fig. 5 shows the total migration time and downtime in VM migration. VMs are migrated in two modes with different working load in them. We can get conclusions from comparison that total migration time and downtime will increase obviously if we use exec mode for migration, especially downtime. Besides, downtime of migration is influenced by different kinds of applications, which will be discussed in Section 6.3. As a self-defined method, the throughput is slightly lower than that of *tcp* mode intrinsically provided in KVM. Thus, migration method selection affects the performance. Difference on total migration is little, but is quite huge on downtime. The discrepancy can be 0.6 to 0.8 second for the two ways. We conduct exec mode VM migration over XIA comparing to that over TCP/IP network.

6.2 Connectivity Test

We test VM migration in *exec* mode over both TCP/IP and XIA networks. Processing programs is required for data sending and receiving with TCP/IP sockets provided in network. A calculation service always runs in the VM that are being migrated for connectivity test. We just have to take one application as a typical example for connectivity test because VM will be unreachable in WAN, no matter which kind of workload it takes along.

First of all, the service runs in VM never get interrupted during the whole migration procedure. Xping that sending ICMP packets to the mobile server drops packets during downtime but recovers soon after VM's resuming on destination host. Both connection-oriented and connection-less service communications can

Table 1. Performance of VM migration in different networks

	Intra-AD		**Inter-AD**	
	total time	downtime	total time	downtime
TCP/IP	6.5 min	1 s	-	-
XIA	14 min	1.2 s	15 min	1.2 s

be recovered during VM migration, even without agent or tunnel mechanisms used for network recovery in WAN.

Total migration time and downtime of above experiments are demonstrated in table 1. We can conclude that full VM migration in LAN of TCP/IP network takes the least total migration time and downtime. When it comes to WAN, the VM and its services are all inaccessible after it has been moved to the destination subnet. The migrated VM has kept the original IP address and this can't be recognized in different subnet.

In our experiments, VM migration can be achieved in XIA network successfully though the performance isn't so good. Downtime is about 0.2s longer than that in TCP/IP while total migration time is about twice longer. The long time is caused by chunk cache mechanism in XIA routers. Chunks that are passing through a router would be cached for future use. The router will search its cache when a chunk request comes and it will deliver this chunk to client if found, or it will continues to forward this request. Thus the time cost for chunk search will sharply increase chunk transmission time. It can also reduce network throughput to some extent. We have implemented some modifications on chunk cache algorithms in XIA, that is, to release the first chunk in the cache table. Actually it turns out to reduce total migration time and downtime by a large margin. Lots of effort should be made for performance optimization in XIA.

6.3 Workload Test

Both of intra-AD and inter-AD VM migrations are implemented with different workloads. We also try to move the VM from one host to another and then back to source site without rebooting so as to match the goal of *"four any"*. Two different routing table modification methods are tested in inter-AD migration. Total migration time and downtime are shown in table 2 and 3 (s: source host, d: destination host).

Calculation application is a workload both CPU-intensive and network-intensive because it calculates results and delivers the data rapidly. *Netperf* and *dbench* are I/O intensive ones. We can conclude from the results that network-intensive workload affects total migration time of VM migration most in XIA. Migration of VMs with I/O intensive workloads suffers longer downtime.

We can find that total migration time decreases slightly when the VM is moving back to the source host. This is mainly caused because of cache hit that has been stored in former rounds of transmission. But the downtime increase a lot during this procedure. We know that the content delivered in the final round is mainly CPU information and hot pages (the pages that are modified frequently). These content chunks have a small chance to be cached on path. But time is still spent on cache search procedures in routers. So the downtime for VM's moving back turns to be much longer. We also find that large quantities of memory and CPU rate are occupied after migration and this will reduce the processing speed of OS. This is another factor that affects downtime, especially when VM is moving back. The factors that cause resources unreleased will be analyzed in our future works.

We also come up with two routing table modification schemes in inter-AD migration. One is chained mode that modification command starts directly from the source host to the router in destination AD. Routers on the path should add a routing entry leading to HID_VM and then pass the command to next-hop router when it receives a modification command. The other is method of aggregation, which means that routing table modification is achieved cooperatively by the command sent from source host and the announcement from the resumed VM. In this procedure, routers on path still finish the task of adding entry and passing command. The difference is that the routing entry on destination AD router is modified according to the information sent out from the resumed VM.

In contrast, the chained method performs better than the aggregation method. In aggregation mode, the routing table entry in the destination AD router is modified after VM is resumed, leading to longer downtime measured. Assume that there are k routers between source and destination AD routers, the time spent on routing table modification is t_1 for each router and the time period form VM's shutdown on source host to its resume on destination host is t_0. Furthermore, if time t_2 is cost for the migrated VM to perceive its mobility and the time for passing a command to next-hop router is t_3. Then the total amount of downtime measured for chained method should be $T_{d1} = max\{(k+2)t_1 + (k+1)t_3, t_0\}$ and the downtime of aggregation method can be expressed as $T_{d2} = max\{(k+1)t_1 + kt_3, t_0 + t_2 + t_3\}$. Actually T_{d2} will be greater than T_{d1} because the time for mobility perception (t_2) is quite long.

Table 2. Intra-AD VM Migration performance in XIA with different workload

	Total time		**Downtime**	
	s->d	d->s	s->d	d->s
Cal	863 s	847 s	1.0 s	2.9 s
Dbench	722 s	718 s	1.2 s	3.3 s
Netperf	726 s	745 s	1.6 s	4.1 s

Table 3. Inter-AD VM migration performance with two self-adaptive methods

	Chained method				**Aggregation method**			
	Total time		**downtime**		**Total time**		**downtime**	
	s->d	d->s	s->d	d->s	s->d	d->s	s->d	d->s
Cal	901 s	888 s	0.8 s	2.8 s	897 s	878 s	0.8 s	3.2 s
Dbench	775 s	750 s	1.1 s	3.2 s	755 s	751 s	1.5 s	3.6 s
Netperf	773 s	762 s	2.1 s	3.6 s	776 s	765 s	1.7 s	4.2 s

7 Conclusion and Future Work

This paper designs experimental testbeds in future Internet prototype XIA for VM migration. Data are transmitted in chunks and this procedure is managed by a migration control protocol. We then introduce a self-adaptive mechanism in order to solve the application interruption problem after migration, especially for migration between ADs. All the traffics directed to VM can be recovered even if they are still using original VM addresses. Evaluation results show that VMs can be moved in XIA networks successfully without application interruption. Performance represented by total migration time and downtime is compelling but needs further optimization.Another research point is migration strategies such as load balancing. It is an interesting topic in VM migration.

Acknowledgments. The authors are sincerely grateful for the technical support from Peter Steenkiste in Carnegie Mellon University and financial support from Shanghai INGEEK Information Technology Co. Ltd.

References

1. Han, D., Anand, A., Dogar, F., et al.: XIA: Efficient Support for Evolvable Internetworking. In: 9th USENIX Conference on Networked Systems Design and Implementation, pp. 23–23. USENIX Association, Berkeley (2012)
2. Anand, A., Dogar, F., Han, D., et al.: XIA: An architecture for an evolvable and trustworthy Internet. In: Proceedings of the 10th ACM Workshop on Hot Topics in Networks, Article No. 2, ACM, New York (2011)
3. Fischer, A., Fessi, A., Carle, G., et al.: Wide-area virtual machine migration as resilience mechanism. In: 30th IEEE Symposium on Reliable Distributed Systems Workshops (SRDSW), pp. 72–77. IEEE Press, New York (2011)
4. Kang, T.S.: Tsugawa. M., Fortes, J., et al.: Reducing the Migration Times of Multiple VMs on WANs Using a Feedback Controller. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), pp. 1480–1489. IEEE, Piscataway (2013)
5. Al-Kiswany, S., Subhraveti, D., Sarkar, P., et al.: VMFlock: virtual machine co-migration for the cloud. In: 20th International Symposium on High Performance Distributed Computing, pp. 159–170. ACM, New York (2011)
6. Comer, D.: A future Internet architecture that supports Cloud Computing. In: 6th International Conference on Future Internet Technologies, pp. 79–83. ACM, New York (2011)
7. Akoush, S., Sohan, R., Rice, A., et al.: Predicting the performance of virtual machine migration. In: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 37–46. IEEE Computer Society, Washington D.C. (2010)
8. Ibrahim, K.Z., Hofmeyr, S., Iancu, C., et al.: Optimized pre-copy live migration for memory intensive applications. In: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. ACM, New York (2011)
9. Michael, R.H., Umesh, D., Kartik, G.: Post-copy live migration of virtual machines. ACM SIGOPS Operating Systems Review 43(3), 14–26 (2009)

10. Luo, Y.W., Zhang, B.B., Wang, X.L., et al.: Live and incremental whole-system migration of virtual machines using block-bitmap. In: IEEE International Conference on Cluster Computing, pp. 99–106. IEEE, Piscataway (2008)
11. Liu, H.K., Jin, H., Liao, X.F., et al.: Live migration of virtual machine based on full system trace and replay. In: 18th ACM International Symposium on High Performance Distributed Computing, pp. 101–110. ACM, New York (2009)
12. Fei, M., Feng, L., Zhen, L.: Live virtual machine migration based on improved pre-copy approach. In: Software Engineering and Service Sciences (ICSESS), pp. 230–233. IEEE, Piscataway (2011)
13. Sato, K., Sato, H., Matsuoka, S.: A Model-Based Algorithm for Optimizing I/O Intensive Applications in Clouds Using VM-based Migration. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 466–471. IEEE Computer Society, Washington D.C. (2009)
14. Piao, J.T., Yan, J.: A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing. In: 9th International Conference on Grid and Cloud Computing, pp. 87–92. IEEE Computer Society, Washington D.C (2010)
15. Liu, H.K., Jin, H., Xu, C.Z.: erformance and Energy Modeling for Live Migration of Virtual Machines. Cluster Computing 16(2), 249–264 (2013)
16. Chen, C., Fan, Y., Zhang, H.X., et al.: A New Live Virtual Machine Migration Strategy. In: International Symposium on Information Technology in Medicine and Education, pp. 173–176. IEEE, Piscataway (2012)
17. Zheng, J., Ng, T.S.E., Sripanidkulchai, K., et al.: COMMA: Coordinating the Migration of Multi-tier Applications. In: 10th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 153–164. ACM, New York (2014)
18. Bradford, R., Kotsovinos, E., Feldmann, A., et al.: Live Wide-Area Migration of Virtual Machines Including Local Persistent State. In: 3rd International Conference on Virtual Execution Environments, pp. 169–179. ACM, New York (2007)
19. Silvera, E., Sharaby, G., Lorenz, D., et al.: IP Mobility to Support Live Migration of Virtual Machines across Subnets. In: SYSTOR 2009, Article No. 13 (2009)
20. Harney, E., Goasguen, S., Martin, J., et al.: The Efficacy of Live Virtual Machine Migrations over the Internet. In: 2nd International Workshop on Virtualization Technology in Distributed Computing, pp. 8–14. ACM, New York (2007)
21. Wood, T., Ramakrishnan, K.K., Shenoy, P., et al.: CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. In: 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 121–132. ACM, New York (2011)
22. Tsugawa, M., Riteau, P., Matsunaga, A.: User-level Virtual Networking Mechanisms to Support Virtual Machine Migration over Multiple Clouds. In: GLOBECOM Workshops, pp. 568–572. IEEE, Piscataway (2010)
23. Kohler, E., Morris, R., Chen, B., et al.: The Click modular router. ACM Transactions on Computer Systems (TOCS) 18(3), 263–297 (2000)

A Near-Exact Defragmentation Scheme to Improve Restore Performance for Cloud Backup Systems

Rongyu Lai, Yu Hua, Dan Feng, Wen Xia, Min Fu, and Yifan Yang

Wuhan National Laboratory for Optoelectronics (WNLO)
Huazhong University of Science and Technology, Wuhan, China
{lairongyu,csyhua,dfeng,xia,fumin}@hust.edu.cn,dickyfyang@gmail.com

Abstract. Cloud backup systems leverage data deduplication to remove duplicate chunks that are shared by many versions. The duplicate chunks are replaced with the references to old chunks via deduplication, instead of being uploaded to the cloud. The consecutive chunks in backup streams are actually stored dispersedly in several *segments* (the storage unit in the cloud), which results in fragmentation for restore. The segments that are referred will be downloaded from the cloud when the users want to restore the chunks of the latest version, and some chunks that are not referred will be downloaded together, thus jeopardizing the restore performance. In order to address this problem, we propose *a near-exact defragmentation scheme*, called NED, for deduplication based cloud backups. The idea behind NED is to compute the ratio of the length of chunks referred by current data stream in a segment to the segment length. If the ratio is smaller than a threshold, the chunks in the data stream that refer to the segment will be labeled as fragments and written to new segments. By efficiently identifying fragmented chunks, NED significantly reduces the number of segments for restore with slight decrease of deduplication ratio. Experiment results based on real-world datasets demonstrate that NED effectively improves the restore performance by 6%∼105% at the cost of 0.1%∼6.5% decrease in terms of deduplication ratio.

1 Introduction

Cloud backup is an important application of cloud storage service. Stefan et al. [9] proposed a data backup system, called Cumulus. Cumulus is a cloud backup system with high portability by only using four basic cloud storage interfaces (GET, PUT, LIST, DELETE) to manage data in the cloud. YuruBackup is a cloud backup system that uses fingerprint servers to obtain highly-efficient deduplication and explores a highly scalable architecture for fingerprint servers to cope with increasing number of clients [12]. Dropbox is a file synchronization tool based on Amazon S3 and it is one of the world's most popular cloud storage applications. In addition, there are many other cloud backup systems, such as Jungle Disk, Brackup.

Some cloud backup systems are designed under a *thin cloud* assumption that the remote data center only provides minimal interfaces (i.e, uploading and downloading complete files). Cumulus [9], Brackup, YuruBackup [12] and Duplicity are this kind of system. The *thin cloud* design ensures that the systems are able to back up data to almost any remote storage. We focus on the restore performance of thin cloud based backup systems in this paper.

In order to improve the backup speed and reduce the storage space, the backup systems employ data deduplication [9], [12] and delta compression [11] due to their salient features of data compression performance. Only deduplication is discussed in this paper. In the backup process, the input chunks are duplicate detected, and duplicate chunks will not be written to the cloud. Instead, the system only keeps references to the stored chunks. Through deduplication, only new chunks will be written to segments and uploaded to the cloud. Therefore, deduplication improves storage utilization and saves backup time for thin cloud based backup systems.

In practice, the deduplication possibly causes degradation of restore performance. Deduplication removes duplicate chunks and keeps references to old chunks stored in the cloud. The deduplication leads to logically consecutive chunks in a data stream not being consecutively stored in *segments* (the storage unit in the cloud). Some referred segments contain lots of chunks that are not referred by the data stream. If the length of chunks referred by a data stream in a segment is smaller than a threshold, the chunks of the data stream that refer to the chunks of the segment are fragments. To restore these fragments, the segments that contain them will be downloaded. Chunks that are not referred in these segments will be downloaded together. But these chunks are useless for the restore and lengthen the restore time, since the segments are downloaded through WAN and the speed of segments reading is much faster than that of segments downloading. The bottleneck in restore is the segments downloading process. Thus the fragments exacerbate the restore performance.

Several defragmentation schemes were proposed to improve restore performance of deduplication based backup systems, such as Capping [5], CFL [6] and CBR [4]. However, they are designed for defragmentation in deduplication-based traditional backup systems. After defragmentation, every duplicate chunk obtains a defragment state, "fragment" or "not fragment". Existing schemes can not guarantee that the duplicate chunks that refer to one segment are in the same state. Some duplicate chunks referring to the chunks of one old segment are possibly determined as fragments and rewritten to new segment. Meanwhile, others are not fragments and keep reference to chunks of the old segment. For restoration, both new segments and the old segments will be downloaded. The fragments will be downloaded twice. Existing schemes mistakenly identify some duplicate chunks to be fragments, and it incurs two main challenges: (1) cloud storage space will be wasted and thus the backup cost will be increased; (2) the backup time will be significantly extended in a low-bandwidth network environment.

In this paper, we propose *a near-exact defragmentation scheme for deduplication based cloud backup (NED)*. It computes the ratio of length of chunks referred by current data stream in a segment to the segment length. If the ratio is smaller than a threshold, the chunks in the data stream that refer to the segment will be labeled as fragments. Thus, the duplicate chunks that refer to a segment will get the same defragment state. If a segment is referred by a fragment, the segment will not be downloaded when the data stream is restored. No chunks will be downloaded twice. Compared with existing defragmentation schemes, NED is able to rewrite fewer chunks than existing schemes while achieving near-exact restore performance of cloud backup system without deduplication. In addition, we need to upload smaller amount of data to the cloud than existing defragmentation, which helps save backup time and storage space.

The rest of this paper is organized as follows. Section 2 presents the previous work of improving restore performance of deduplication based backup system. The background and motivation of our work are proposed in section 3. In section 4, we present design and implementation of NED. Section 5 presents and discusses the experiment results of NED. Finally, section 6 concludes the paper.

2 Related Work

Researchers have proposed some schemes to improve restore performance in deduplication based storage systems. These schemes are divided into two categories according to the principles, *"defragmentation"* and *"deduplication"*. iDedup [7], CFL [6], CBR [4], and Capping [5] are *"defragmentation"* schemes that aim to identify and rewrite the fragments. CABdedup [8] is a *"deduplication"* scheme, and it employs data deduplication in the restore process.

iDedup is a deduplicaiton solution for primary workloads which are latency-sensitive. iDedup efficiently reduces the latency of deduplication, but it is not appropriate for cloud backup because it rewrites too much data and cloud backup workloads are not latency-sensitive in the deduplication stage.

CFL-SD selectively deduplicates the input chunks based on chunk fragment level (CFL) to improve restore performance. CBR rewrites the fragmented chunks based on stream context and disk context to improve restore performance. Capping improves restore performance by analyzing the fragmented chunks in a much larger buffer of input chunks. Specifically, Capping inserts chunks to a fixed-length (for example, 20MB) buffer, and computes the count of segments referred by the chunks in the buffer. Capping only deduplicates the chunks in the top 10 referred segments and rewrites other chunks for defragmentation.

CABdedup points out that deduplication not only can be applied to improve backup speed and save storage space in the backup process, but also can be used in the restore process. However, it doesn't address the fragmentation problem caused by deduplication.

Some cache techniques on chip-multiprocessors [15], [14], [13] are used to speed up the fingerprint computing, which helps improve the backup speed. However, the restore doesn't consume much computing time, and the techniques don't have obvious effect on restore.

Table 1. Table of related work. This table shows how *NED* is positioned relative with some other relevant work.

Name	Usage Scenario	Design Goal & Solution
iDedup [7]	Primary storage	It executes selective deduplication to improve restore performance of deduplication system
CFL-SD [6]	Traditional backup	It executes defragmentation based on chunk fragment level to improve restore performance of deduplication system
CBR [4]		It executes defragmentation based on stream context and disk context to improve restore performance of deduplication system
Capping [5]		It executes defragmentation based on buffer analysis to improve restore performance of deduplication system
CABdedupe [8]	Cloud storage	It executes deduplication in the restore process to improve restore performance of deduplication system
NED		It executes defragmentation based on segment reference analysis to improve restore performance of deduplication system

Table 1 summarizes the above restore performance optimization schemes. There is not a *"defragmentation"* scheme for cloud backup. We analyze the reason in detail in section 3. Therefore, we propose NED to remove fragments in deduplication based cloud backup systems.

3 Background and Motivation

3.1 Cloud Backup

Some nomenclatures are explained below to help describe how cloud backup system works clearly.

Data set and data stream. A data set includes many versions of file collection. Files in a collection are divided into chunks, and the chunks form a data stream. For example, chunks A $\sim$ L form a data stream in Figure 2.

Segment. Segment is the storage unit in the cloud [9], [8], [12]. Cloud backup systems write chunks into larger units called segments. The chunk that makes a segment be longer than the specified maximum segment (4MB default) will be written to a new segment, and the previous segment will be closed. A segment is identified by a segment ID.

To get high portability, the cloud backup systems only employ simple interfaces (e.g, get, put, delete, list etc). They don't depend on the ability to read or write arbitrary byte ranges within a file. The interfaces are simple enough that they can be implemented on various protocols, such as Amazon S3, FTP, SFTP and network file systems [9]. Thus, a complete segment is uploaded and downloaded.

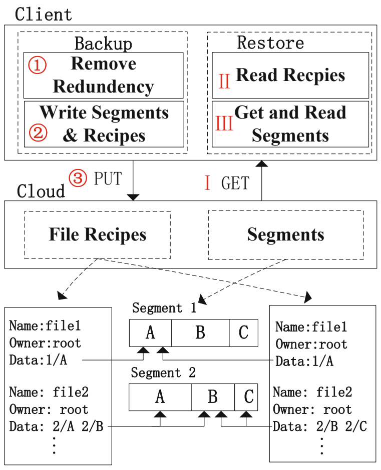


Fig. 1. The backup and restore process

Recipe. Each job has a recipe. The main contents of a recipe are the list of files that are backed up in the job and the chunks' addresses of the files. In addition, it contains job information, such as backup time. Figure 1 shows two simplified recipes. Recipes are written for recovery jobs. The client gets the addresses of chunks from the recipe and gets segments from remote storage according to the addresses.

Backup process. The left part of Figure 1 shows the 3 steps of backup. The first step is removing redundancy which includes dividing files into chunks, computing fingerprints and searching fingerprints. This step is the same as that of traditional backup system (we treat backup system that doesn't use cloud storage as traditional backup system). The fingerprint searching is done in the client or in the metadata server [10]. The system writes new chunks to segments and writes chunk addresses to the recipe in the second step. In the third step, the recipe and segments are uploaded to the cloud.

Restore process. The right part of Figure 1 shows the restore process of the cloud backup system. The restore process includes 3 steps. First, the client downloads the recipe of the job from the cloud. Second, the chunk addresses (segment ID and chunk ID) are read from the recipe. Third, segments are downloaded from the cloud according to the addresses and chunks are read from the segments to construct the files. However, a segment is possibly referred by multiple files, and repetitive downloading of segments severely slows the restore speed when the system restores different files. Therefore, the client employs a buffer on the disk to save the segments that are already downloaded. When a segment is needed, the client checks the buffer. If the segment already exists in the buffer, the client reads it directly from the buffer. Otherwise, the client downloads the segment from the cloud and then inserts it to the buffer. Thus, a segment will be downloaded from the cloud only once.

Table 2. The comparison between cloud backup and traditional backup

	Cloud Backup	Traditional Backup
General User	Individual	Enterprise
Size of Backup Set	Small	Big
Bandwidth	Low	Hight
Storage Location	Cloud	Data Server
Storage Unit	Segment	Container
The Bottleneck of Restore	WAN	Disk
Representative Systems	Cumulus[9], Dropbox Jungle Disk, Brackup duplicity, YuruBackup[12]	HydraStor[1], DDFS[16] Symantec[3] GreenBytes

Table 2 summarizes the comparison between cloud backup and traditional backup. In traditional backup, chunks are sent to the client after being read from the containers in the data servers in restore process[2]. If a data server is running multiple jobs, reading data from disk will become a bottleneck in traditional backup. Since thin cloud system employs interface of reading a complete file (i.e., segment)[9], data is read in the client after the segments are downloaded from the cloud in cloud backup [12]. In general, the speed of reading data from disk is much faster than that of translating data through WAN. Therefore, the bottleneck of restore process in cloud backup is the segment downloading process.

3.2 Definition of Fragment in Cloud Backup System

Chunks are shared by new version and old version after deduplication. The system replaces duplicate chunks with the references to old chunks instead of uploading the chunks to the cloud. The consecutive chunks in backup stream are actually stored dispersedly. The lengths of referred chunks in some segments are possibly longer than those of other segments. In order to accurately measure the length of data referred by the data stream in a segment, we introduce two new terms, *segment reference length (SRL)* and *segment reference ratio (SRR)*. SRL is the total length of data referred by a data stream in a segment, and SRR is the ratio of SRL to the length of the segment. Due to the different distributions of data streams, the SRR of a segment differs in different data streams. If a duplicate chunk is restored in cloud backup system, the complete segment that is referred must be downloaded. Some chunks that are not referred by any chunks of the data stream will possibly be downloaded together.

Segment reference ratio threshold is proposed to help the system identify fragments. If a duplicate chunk refers to a segment whose reference ratio is smaller than the segment reference ratio threshold, the chunk is a fragment. For example, we assume that the threshold is 0.3. In Figure 2, the reference ratio of segment 37 is 0.75, and it is bigger than the threshold. Therefore,

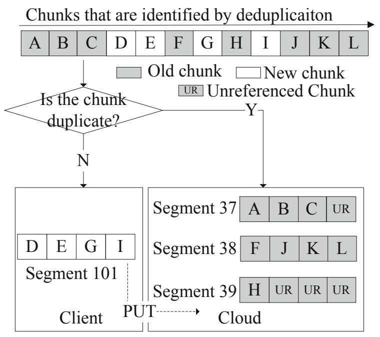


Fig. 2. An example of fragment in cloud backup

chunks A, B and C are not fragments. The same are with chunks F, J, K and L. Segment 39's reference ratio is 0.25 which is smaller than the threshold. Therefor, chunk H is a fragment.

3.3 Motivation

As shown in Table 1, there is not a defragmentation scheme for cloud backup. Existing defragmentation schemes mistakenly identify fragments in cloud backup. After defragmentation, every duplicate chunk gets a "fragment" or "not fragment" state. The existing schemes are not able to ensure that the duplicate chunks that refer to one segment are in the same state. Part of duplicate chunks that refer to an old segment are identified to be fragments and rewritten to new segment, and some other duplicate chunks that refer to the same segment are not fragments and keep the references. Both the new segment and the old segment will be downloaded in the restore process, and the fragments will be downloaded twice.

This sort of rewriting not only wastes the cloud storage space and lengthens the backup time, but also jeopardizes the restore performance. This is demonstrated in Section 5.5 and 5.6. A defragmentation scheme for deduplication based cloud backup makes sense.

4 Design and Implementation

4.1 Near-Exact Defragmentation Scheme for Deduplication Based Cloud Backup

In order to identify and rewrite the fragments in cloud backup, we propose *a near-exact defragmentation scheme for deduplication based cloud backup (NED).* NED is designed as an independent module and provides simple input and output interfaces. Thus, NED can be conveniently added to deduplication systems.

As Section 3.1 describes, the backup process of a typical cloud backup system includes 3 steps. The duplicate chunks are identified in the first step. After the first step, every chunk gets a deduplication state, "duplicate" or "new". The second step includes writing new chunks to segments and writing the recipe. The recipe and the segments are uploaded to the cloud in the third step. NED works after the first step and before the second step.

Input. The input of NED includes chunk contents and chunk information. The chunk information contains chunk length, deduplication state, chunk address in local file system and duplicate chunk's reference.

Output. The output of NED includes not only the defragment state, but also chunk information and chunk contents that are inputted. Thus, the output keeps consistency with the input, and NED can be added to the system conveniently.

NED is constructed with *Chunk Dedup Result Buffer (CDRB)*, *Segment Reference Buffer (SRB)* and *Rewriting Monitor (RM)*. The chunk information is stored in CDRB. ID of referred segment, the corresponding *Segment Reference Length (SRL)* and *Segment Reference Ratio (SRR)* are stored in the records of SRB. Fragments are identified by RM.

4.2 Workflow

Workflow of NED is divided into *segment reference ratio statistics phase* and *defragment phase*. The defragment phase starts after all the chunks in the data stream are inputted into NED. Figure 3 shows the NED workflow.

Segment reference ratio statistics phrase. After a chunk is inputted into NED, NED inserts the chunk information into CDRB, and checks the deduplication state of the chunk. If the chunk is duplicate, NED searches SRB for the right record by the the segment ID of the chunk reference (segment ID & chunk ID). A match occurs if the segment ID in the record is equal to the segment ID of the chunk reference. If no match occurs, NED creates a new record. NED updates the SRR in the record, and the updating operation needs to add the chunk length to the SRL and compute the SRR. Because the defragment phase doesn't start until all the chunks in the data stream are inputted into NED, all inputted chunks will be stored in CDRB. However, the chunk contents are inputted into NED together with the chunk information, and the RAM consumption will be large if all the chunk contents are stored in RAM. To address this problem, NED frees the chunk contents to save RAM space.

Defragment phase. The goal of this phase is to identify the fragments within the chunks that are stored in the CDRB. Before discussing this phase, we assume that the segment reference ratio threshold is p. RM traverses all the chunks in the CDRB. If there comes a duplicate chunk, RM searches the SRB for the corresponding record by the segment ID of the chunk's reference. The chunk will be marked as a fragment if the SRR in the matched record is smaller than p. In order to keep the RAM overhead as low as possible, the chunk contents were freed at the first phase. In order to keep the format of output compatible with the input and meet the requirements of underlying storage, NED reads the chunk contents from local file system according to the local file system addresses of chunks in this phase.

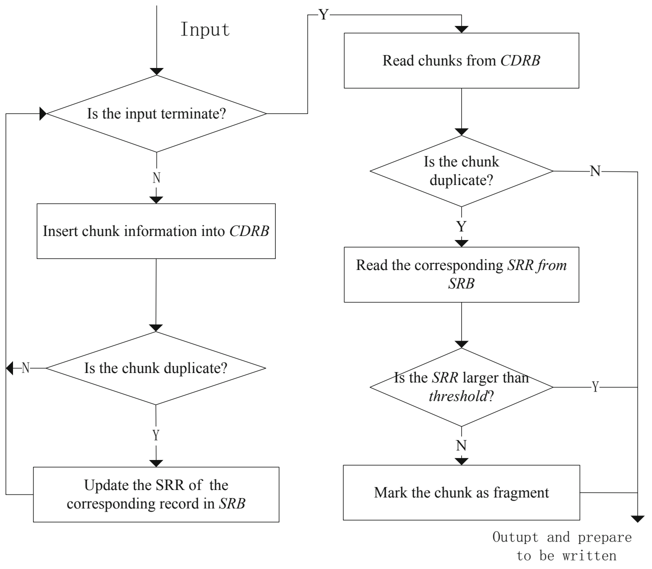

Input
Is the input terminate?
Y
N
Insert chunk information into CDRB
Is the chunk duplicate?
N
Y
Update the SRR of the corresponding record in SRB
Read chunks from CDRB
Is the chunk duplicate?
N
Y
Read the corresponding SRR from SRB
Is the SRR larger than threshold?
Y
N
Mark the chunk as fragment
Outupt and prepare to be written

Fig. 3. The workflow of NED

As shown in Figure 4, we illustrate the working process of cloud backup system which employs NED. Chunks identified by deduplication are inputted into NED module. At segment reference ratio statistics phase, the coming chunks are sequentially inserted into CDRB and SRRs of referred segments in SRB are updated at the same time. After the first phase, segments 37, 38, and 39 are referred and the SRRs are 0.75, 1, 0.25 respectively. We assume that segment reference ratio threshold is 0.3. At the defragment phase, RM successively obtains chunks A∼L from CDRB, and determines whether they are fragments or not. Chunks A, B and C are duplicate chunks and they refer to old chunks in segment 37. The reference ratio of segment 37 is bigger than the threshold. Therefore, chunks A, B and C are not fragments. Similarly, chunks F, J, K and L are not fragments. The SRR of segment 39 is smaller than the threshold, and chunk H is a fragment.

NED analyzes the entire data stream. Duplicate chunks that refer to one segment will get the same defragment state. As the experiment results show, when the threshold is 0.9, NED near-exactly identifies the fragments in the data stream and the average restore performance is almost equal to that of backing up without deduplication.

5 Performance Evaluation

5.1 Experimental Setup

In order to test the performance of NED, we develop a simulator which is a 5,000-line C program. The simulator employs rabin fingerprint algorithm for chunking,

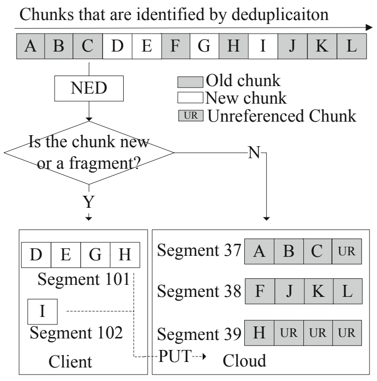


Fig. 4. An example of NED defragmentation

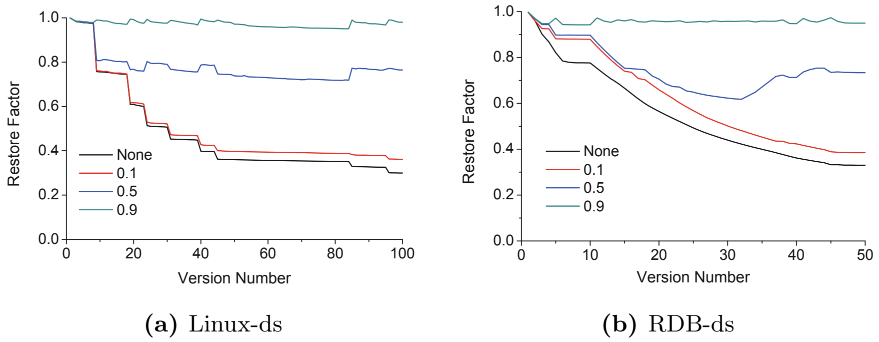


Fig. 5. The restore performance. *None* denotes the no defragmentation mode, and *0.1*, *0.5* and *0.9* are the segment reference ratios of NED.

and the average chunk length is 8KB. It computes fingerprints via SHA-1. The maximal segment length is 4MB. The simulator employs hash table to store fingerprints. We implement the simulator on a machine running Ubuntu 12.04 and the machine is featured with Intel i7-930 CPU @2.80GHz, 2TB 7200RPM hard drive. We use two data sets as workload.

- Linux-ds. They are the Linux source files, from Linux 2.6.34 to Linux 3.2.04 total of 100 versions, and each version has more than 30,000 small files. The average total length of each version is 400MB. The total size of the data set is about 40GB.
- RDB-ds. We get the second data set from daily backup of Redis database for 50 days. Each version includes a mirror file whose average size is 5GB. The total size of the data set is about 250GB.

5.2 Performance Measurement

We propose *restore factor* and *average restore factor* to measure the restore performance of cloud backup system, and propose *deduplication ratio* to measure the deduplication efficiency.

Restore factor is 1/the length of segment downloaded per MB of data restored. As described in section 3.2, the bottleneck of restore in cloud backup is the segment downloading process. Less data downloaded per MB data restored indicates higher restore performance. Therefore, we propose restore factor to measure the restore performance of a single version. The larger the restore factor is, the higher the restore performance is.

Average restore factor is the average restore factor of many versions. It is used to measure the average restore performance of multiple versions.

Deduplication ratio is the ratio of the length of removed duplicate data to the length of the data stream. Only the compression caused by deduplication is considered in this paper.

5.3 The Restore Performance

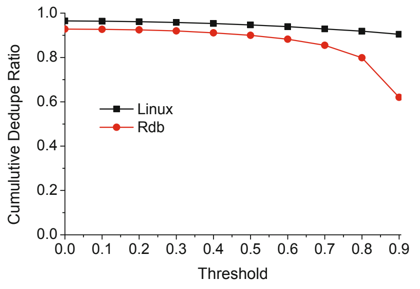
Figure 5 shows the impact of NED on restore performance. The baseline is the restore performance of no defragmentation mode. When the threshold is 0.1, restore performance is slightly improved. The restore performance is approximately 2-fold to the baseline when the threshold is 0.5. When threshold is 0.9, NED near-exactly removes the fragments. Restore factor nearly reaches 1 and is far higher than the baseline. Bigger threshold indicates that more chunks are identified to be fragments. NED significantly improves the restore performance when the threshold is bigger than 0.5.

5.4 The Segment Reference Ratio Threshold

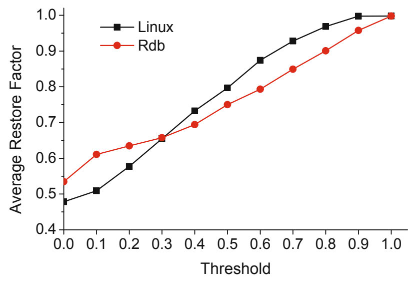
Figure 6 shows the effect of varying segment reference ratio threshold on deduplication ratio and restore factor. No duplicate chunk is identified to be fragment and rewritten when segment reference ratio threshold is 0. Thus, the duplication ratio is the biggest and the restore factor is the smallest. All duplicate chunks are identified to be fragments when the threshold is 1, and the restore factor is the biggest. Larger threshold indicates that more duplicate chunks are marked as fragments. The increase of threshold improves the restore performance and decreases the deduplication ratio. With the growth of threshold, the restore factor increases by 6%∼105%, and deduplication ratio decreases by 0.1%∼6.5% in Linux-ds. In RDB-ds, the restore performance increases by 12%∼75%, and deduplication ratio decreases by 0.1%∼32.5%.

5.5 The Relationship of Restore Factor and Deduplicaton Ratio

Figure 7 shows the relationship between restore factor and deduplication ratio. Bigger restore factor indicates smaller deduplication ratio. Capping is tested as

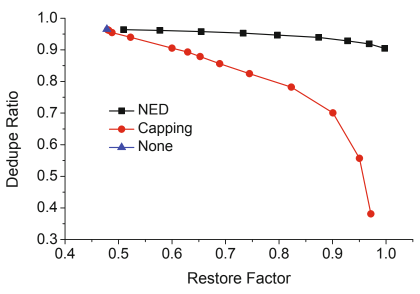


(a) Dedup Ratio

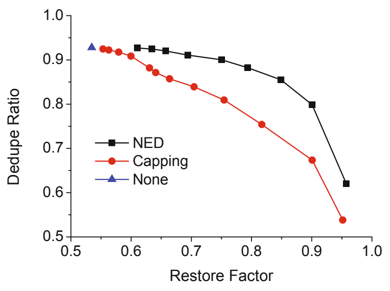


(b) Restore Factor

Fig. 6. Effect of varying segment reference ratio threshold on deduplication ratio and restore factor. The deduplication ratios and restore factors are the average values of the last 10 backups



(a) Linux-ds



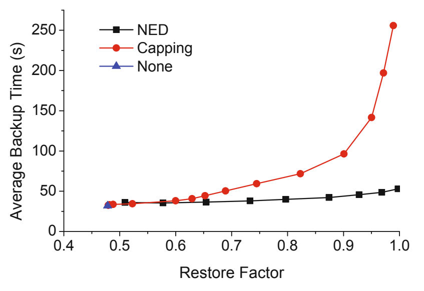
(b) RDB-ds

Fig. 7. The relationship of restore factor and deduplication ratio. *None* denotes no deduplication mode. The deduplication ratios and restore factors are the average values of the last 10 backups.

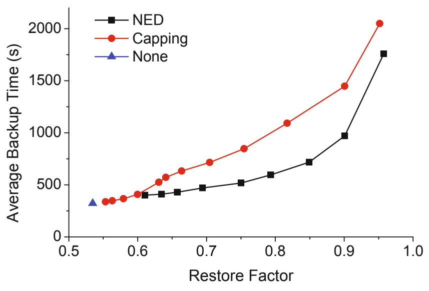
an example of existing defragmentation schemes. In both data sets, the deduplication ratios of NED are larger than those of Capping in the same restore factors. It demonstrates the observation in section 3.3 that the existing defragment schemes mistakenly determine fragments in cloud backup.

5.6 Backup Time

Figure 8 shows the average backup time of different defragment schemes. From the figure we can see, both Capping and NED bring time overhead. The backup time of NED is shorter than that of Capping in the same restore performance, and the gap between them grows with the restore factor. NED spends more time in identifying fragments than existing schemes, but it identifies fragments more accurately. The deduplication ratio of NED is bigger than that of Capping in the same restore factor. That means fewer chunks are uploaded and the uploading time is shorter. In low-bandwidth WAN environment, the reduced uploading

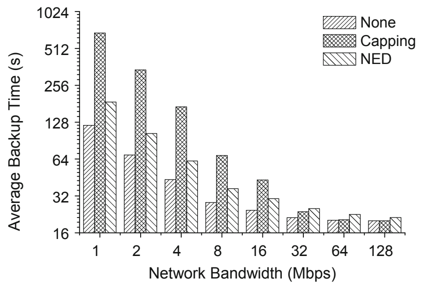


(a) Linux-ds

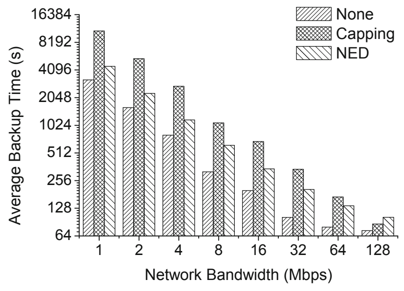


(b) RDB-ds

Fig. 8. The time overhead. *None* denotes no deduplication mode. The restore factors are the average values of the last 10 backups.



(a) Linux-ds



(b) RDB-ds

Fig. 9. The average backup time in different WAN bandwidths. The threshold of NED is 0.55, and Capping level is 5 segments per 20MB. With this setup, both NED and Capping improve the average restore factor to about 0.82.

time is actually much longer than the defragment time of NED. The result is that the backup time of NED is much shorter than the existing schemes.

As shown in Figure 9, lower bandwidth indicates wider uploading gap between NED and Capping. Because Capping rewrites more data than NED in the same restore factor.

NED has different effects on restore performance and deduplication ratio in different workloads. The users should adjust the segment reference ratio to make sure that not only the restore performance meets the requirements but also the decrease of deduplication ratio is acceptable. The restore performance is improved greatly when the threshold is 0.5, and the deduplication ratio decreases slightly. 0.5 is recommended as the default threshold.

6 Conclusion

Due to chunk fragmentation in deduplication based cloud backup systems, the restore performance becomes worse when the version number grows. However, existing defragmentation schemes fail to identify fragments in cloud backup. We propose NED to identify fragments in cloud backup by precisely indentifying fragmented chunks in each backup. The experiment results show that NED effectively improves the restore performance at the cost of limited decline in terms of deduplication ratio.

Acknowledgment. This work was supported in part by National Natural Science Foundation of China (NSFC) 61173043; National Basic Research 973 Program of China 2011CB302301; NSFC 61025008, 61232004; the Seed Project of Wuhan National Laboratory for Optoelectronics (WNLO).

References

1. Dubnicki, C., Gryz, L., Heldt, L., Kaczmarczyk, M., Kilian, W., Strzelczak, P., Szczepkowski, J., Ungureanu, C., Welnicki, M.: Hydrastor: A scalable secondary storage. In: Proccedings of the 7th Conference on File and Storage Technologies, pp. 197–210. USENIX (2009)
2. Fu, M., Feng, D., Hua, Y., He, X., Chen, Z., Xia, W., Huang, F., Liu, Q.: Accelerating restore and garbage collection in deduplication-based backup systems via exploiting historical information. In: 2014 USENIX Annual Technical Conference (USENIX ATC 14), pp. 181–192 (2014)
3. Guo, F., Efstathopoulos, P.: Building a high-performance deduplication system. In: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference. USENIX (2011)
4. Kaczmarczyk, M., Barczynski, M., Kilian, W., Dubnicki, C.: Reducing impact of data fragmentation caused by in-line deduplication. In: Proceedings of the 5th Annual International Systems and Storage Conference, pp. 15:1–15:12. ACM (2012)
5. Lillibridge, M., Eshghi, K., Bhagwat, D.: Improving restore speed for backup systems that use inline chunk-based deduplication. Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 2013), pp. 183–197. USENIX (2013)
6. Nam, Y.J., Park, D., Du, D.H.C.: Assuring demanded read performance of data deduplication storage with backup datasets. In: Proceedings of the 2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 201–208. IEEE (2012)
7. Srinivasan, K., Bisson, T., Goodson, G., Voruganti, K.: idedup: Latency-aware, inline data deduplication for primary storage. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies. USENIX (2012)
8. Tan, Y., Jiang, H., Feng, D., Tian, L., Yan, Z.: Cabdedupe: A causality-based deduplication performance booster for cloud backup services. In: Parallel & Distributed Processing Symposium (IPDPS), pp. 1266–1277. IEEE (2011)
9. Vrable, M., Savage, S., Voelker, G.M.: Cumulus: Filesystem backup to the cloud. Trans. Storage 5(4), 14:1–14:28 (2009)
10. Xia, W., Jiang, H., Feng, D., Hua, Y.: Silo: A similarity-locality based near-exact deduplication scheme with low ram overhead and high throughput. In: USENIX Annual Technical Conference (2011)
11. Xia, W., Jiang, H., Feng, D., Tian, L.: Combining deduplication and delta compression to achieve low-overhead data reduction on backup datasets. In: Data Compression Conference (DCC 2014), pp. 203–212 (2014)

12. Xu, Q., Zhao, L., Xiao, M., Liu, A., Dai, Y.: Yurubackup: A space-efficient and highly scalable incremental backup system in the cloud. International Journal of Parallel Programming, 1–23 (2013)
13. Zhan, D., Jiang, H., Seth, S.: Exploiting set-level non-uniformity of capacity demand to enhance cmp cooperative caching. In: 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), pp. 1–10 (2010)
14. Zhan, D., Jiang, H., Seth, S.: Stem: Spatiotemporal management of capacity for intra-core last level caches. In: 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 163–174 (2010)
15. Zhan, D., Jiang, H., Seth, S.C.: Locality & utility co-optimization for practical capacity management of shared last level caches. In: Proceedings of the 26th ACM International Conference on Supercomputing, pp. 279–290. ACM (2012)
16. Zhu, B., Li, K., Patterson, H.: Avoiding the disk bottleneck in the data domain deduplication file system. In: Proceedings of the 6th USENIX Conference on File and Storage Technologies, pp. 18:1–18:14. USENIX (2008)

A Music Recommendation Method for Large-Scale Music Library on a Heterogeneous Platform

Yao Zheng[1,2,3,*], Limin Xiao[1,2,*], Wenqi Tang[1,2], and Li Ruan[1,2]

[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[2] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[3] Aviation Institute, Beijing 101121, China
zyshren@cse.buaa.edu.cn, xiaolm@buaa.edu.cn

Abstract. Currently, music recommendation system is a research focus in music information retrieval and a typical system can handle millions of music in real time. However, online music libraries have exceeded ten-million magnitudes, such as Amazon MP3, which results in mismatching between music recommendation systems and music libraries. Thus, this paper presents a music recommendation method for retrieving the large-scale music library on a heterogeneous platform. Based on the music similarity algorithm, by combining the indexing mechanism with GPU hardware acceleration, we further enhance the processing scale of the proposed method. Experiments show that, without lowering the retrieval accuracy, the proposed music recommendation method has the ability to handle ten-million magnitude libraries online in a single server.

Keywords: Music recommendation, Music similarity algorithm, Large-scale, GPU-accelerated.

1 Introduction

Recently, along with the rapid development of multimedia and Internet technology, the digital music has entered the public life in many aspects, and is influencing the public's media consumption habits in essence. At the same time, we are expanding the size of digital music database. Up to June 2013, the size of the Amazon online music store has surpassed 20 million [1]. Facing the massive digital music, how to help users find the target songs effectively and quickly is the main task of modern music recommendation system. Therefore, the personalized music recommendation system is becoming a research hotspot in the field of music information retrieval (MIR).

Music recommendation system is usually divided into three types: content-based, collaborative filtering and hybrid [2]. Since the collaborative filtering method requires a large amount of historical data sets, there are the sparse and cold start problems. In

[*] Corresponding author.

this paper, we focus on the content-based music recommendation system, which first extracts the underlying characteristics of songs by analyzing audio signal, and then builds a playlists of similar songs into playlists. This method has high computational complexity, which limits the scale of music library. When we gradually increase the size of music library, in particular, to tens of millions of magnitude as Amazon's online music store, the typical content-based music recommendation algorithms have been unable to meet the user's search request. Thus, we need to use the appropriate accelerated methods to recommend the list within the acceptable time.

Scholars have conducted a lot of research in terms of music recommendation algorithm acceleration. There are early indexing methods for vector features [3] and the local sensitive hash (LSH) method [4], whose can only deal with vector feature representation. With the music recommendation algorithms progress and the emergence of the non-vector music features (such as Gaussian models), the above methods will lose its effect. FastMap can convert the non-vector features into vector features, which also can handle a mega music library [5]. However, the simply FastMap approach is unable to meet user requirements against the order of ten million music library.

GPU is a general-purpose many-core computing platform, which has demonstrated significant speedups for many scientific applications, including music information retrieval (MIR). In this paper, we address the problem of handled sizes limitation for the music recommendation approach on a heterogeneous platform with a GPU. We propose a GPU-based music recommendation approach to further enhance the processing scale by combining the indexing mechanism with GPU hardware acceleration.

The rest of the paper is organized as follows: In Section II, preliminary concepts for the music recommendation system and music similarity algorithms were described briefly. Section III introduces the music similarity algorithm, which is the foundation of our proposed method. Section IV presents a GPU-based music recommendation method, and the CUDA implementation of the proposed method is given, which mainly focuses on exploiting acceleration on many-core GPU architectures. In Section V, experimental results are presented to demonstrate the performance of the proposed method. The last section concludes the whole paper and points out some future works.

2 Related Works

The music recommendation system for large-scale music library focuses on how to scan the library quickly, which usually uses the corresponding index mechanism to improve scanning efficiency. Scholars in this area have done lots of works. Traditional indexing method, such as local sensitive hash (LSH), belongs to a high-dimensional data indexing method, which is more suitable for high-dimensional music feature vectors. And researchers have conducted in-depth study of this aspect. Meanwhile, with the advances of music similar algorithms, Non-vectorized musical feature representation is becoming the trend. Hence, the indexing method of non-vectorial musical features is a hotspot. In addition, some researchers consider hardware platforms to speed up the music retrieval, which provides a new idea for the development of large-scale music recommendation system.

2.1 Indexing Method Based on LSH

LSH-based indexing method [6] is an approximate nearest neighbor indexing method. Casey [7] uses LSH to extend its musical similarity algorithm. The algorithm uses two feature vectors to improve the retrieval accuracy, while using LSH to improve the retrieval speed. However, the music library only contains 2,000 songs, which cannot be used to verify the performance of large-scale changes. Cai [8] uses 32-dimensional feature vectors and LSH to establish similar music retrieval. The feature vector combines timbre and temporal changes, but the method is not suitable for non-vector features. Furthermore, the retrieval efficiency decreases with increasing the size of the library. For instance, when the scale reaches 10^5, the average search time is 2.53s. In summary, non-vectorial musical feature representations, such as Gaussian models and Hidden Markov models, are not suitable to build LSH indexing; meanwhile, high computational cost of LSH also hinders its used for large-scale music library.

2.2 Indexing Method for Non-vectorial Musical Feature

Currently, non-vectorial music feature representation is the trend, such as the Gaussian timbre model[23]. Hence, how to build an index approach for Gaussian timbre model is a hotspot. Roy et al. [9] presented a music recommendation system which could be used for large databases using Gaussian timbre features. They use a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence to measure music similarity. This method is far more expensive to compute and yields worse results [10]. Levy and Sandler [11] propose to use Gaussians with a diagonal covariance matrix, instead of a full one to compute music similarity. They report a ten-fold speedup compared to the full KL divergence. However, the quality of this simpler similarity measure is degraded. A number of general methods from the class of distance-based indexing methods are relevant for indexing Gaussian features. Yianilos et al. [12] use vantage-point (VP) Trees to build a tree structure that can be searched efficiently, but require a metric distance measure. Athitsos [13] uses FastMap [14] to randomly embed arbitrary features in Euclidean space and uses LSH to index that mapping. Cano et al. [15] use FastMap to map the high dimensional music timbre similarity space into a 2-dimensional space for visualization purposes.

2.3 Hardware-Based Acceleration Method

With the continuous increase of music data, scholars propose to use parallel processing capabilities of hardware platforms to improve retrieval speed [27-28]. Poil [16] accelerated voice command recognition with dynamic time warping (DTW) by using CUDA, which improved the performance by 75%. Pascal [17] presented a GPU-based Query-By-Humming method, which achieved 162x speed ratio. Adriana

[18] proposed a GPU-accelerated audio feature extraction algorithm, which computed the cross correlation on GPU. Furthermore, FPGA, clusters and distributed systems also have been used to accelerate the music retrieval.

3 Music Similarity Algorithm

Music similarity algorithms usually operate with content-based music recommendation systems, and the recommendation systems work as a query-by-example system: (i) the user selects a favorite song, (ii) the system searches its databases for similar songs, (iii) according to the similarity measure the k-NN are returned to the user as possible recommendations. Hence, music similarity algorithm is the core of music recommendation system. With the improvement of music similarity algorithms, non-vectorial representation, such as Gaussian Models, is used to represent a song, instead of vector representation. Table 1 lists top-ranked algorithms of the MIREX AMSR evaluation [19].

Table 1. Comparison of music similarity algorithms

Year	Algorithm	Music Representation	Similarity Function	Flops
2005	ME[20]	multivariate Gaussians	symmetrized Kullback-Leibler divergence (SKL)	3552
2006	EP [21]	multivariate Gaussians Fluctuation Patterns	$\sqrt{SKL}$	4636
2009	PS [22]	multivariate Gaussians Fluctuation Patterns	Jensen-Shannon approximation	35223

As shown in Table 1, the computational complexity of algorithms is increasing as the accuracy improvement. Compromise between accuracy and computational complexity, we select EP as the basic algorithm for this study. Meanwhile, PS is an improved version of EP, which indicates that EP has good scalability.

EP combines the single Gaussian musical timbre features of ME with the fluctuation patterns to include rhythm information in the similarity measure. The fluctuation patterns are computed from the Mel spectrum of a song. There are three main procedures in EP algorithm, which are initialization and constant calculation, melody library feature extraction and melody similarity calculation.

In the procedure of initialization and constant calculation, we mainly get some constants. These constants remain unchanged in the whole process of EP algorithm. Thus we can calculate these constants once and store them into files for future use.

In the procedure of melody library feature extraction, there are three steps, power spectrum, fluctuation patterns and single Gaussian. These steps are mutually independent but have logical relationships. Calculating fluctuation patterns and single Gaussian needs the result of power spectrum.

In the procedure of melody similarity calculation, a distance matrix D with the size of n×n is calculated between files in the melody library. To calculate D, we must go through all its elements.

4 GPU-Accelerated Music Recommendation Method

Based on the EP algorithm, we propose a music recommendation method for large-scale library, which combines indexing and hardware-accelerated to improve the processing scale.

As shown in Fig.1, the method includes two parts, online and offline. The offline part mainly establishes melody feature library and feature indexing library. The online part includes GPU-based coarse filter for feature vectors and similarity computation for candidate songs.

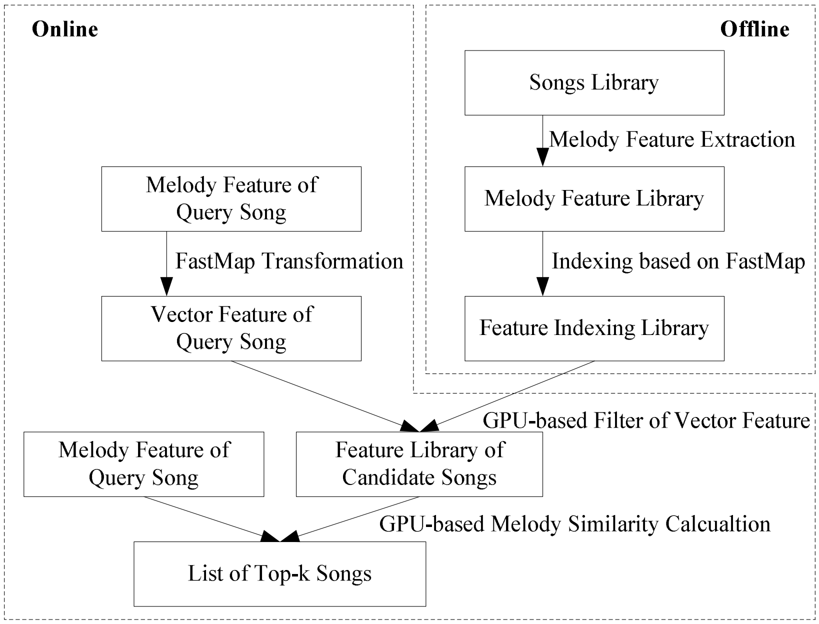

Online
Offline
Songs Library
Melody Feature Extraction
Melody Feature Library
Indexing based on FastMap
Feature Indexing Library
Melody Feature of Query Song
FastMap Transformation
Vector Feature of Query Song
GPU-based Filter of Vector Feature
Melody Feature of Query Song
Feature Library of Candidate Songs
GPU-based Melody Similarity Calcualtion
List of Top-k Songs

Fig. 1. GPU- Accelerated Music Recommendation Method

4.1 GPU-Accelerated Filter for Feature Vectors

In order to decrease the melody similarity calculation, we transform the non-vectorial musical features into k-dimensional vector features using FastMap, and then we calculate the similarity of vector features using Euclidean distance. At last, based on the filtering parameter *fs*, we implement a coarse filter of the melody feature library.

The FastMap algorithm uses a simple mapping formula (Equation (1)) to compute a k-dimensional projection of objects into the Euclidean vector space.

$$F_i(x) = \frac{d(x, x_{i,1})^2 + d(x_{i,1}, x_{i,2})^2 - d(x, x_{i,2})^2}{2d(x_{i,1}, x_{i,2})} \tag{1}$$

where, *d*() represents the distance metrics of non-vectorial musical features.

To project the musical feature into a k-dimensional Euclidean vector space, first two pivot features have to be selected for each of the *k* dimensions. The algorithm uses a simple random heuristic to select those pivot features: for each dimension ($i = 1 \ldots k$), (i) choose a random feature x_i from the database, (ii) search for the feature most distant from x_i using the distance measure *d()* and select it as the first pivot feature $x_{i,1}$ for the dimension, (iii) the second pivot feature $x_{i,2}$ is the object most distant to $x_{i,1}$ in the original space. After the 2k pivot features have been selected, the vector representation of is computed by calculating $F_i(x)$ for each dimension ($i = 1 \ldots k$).

By analyzing the similarity calculation of vector feature library, we found that the calculation belongs to compute-intensive tasks, and has no data dependencies between tasks, which fully meet the requirements of GPU acceleration.

Thus, we propose a GPU-accelerated filter for feature vectors. First, we use one thread on GPU to calculate one distance computation of feature vectors. Then, we accelerate the distance sort on GPU. Finally, the candidate songs are listed based on *fs*.

4.2 GPU-Accelerated Melody Similarity Calculation

Based on the candidate songs, we calculate the melody similarity. In the procedure of melody similarity calculation, a distance matrix D with the size of nxn is calculated between files in the melody library. To calculate D, we must go through all its elements. So we use one thread on GPU to calculate one element in matrix D by starting up nxn threads in total. The correspondence is shown in Fig.2.

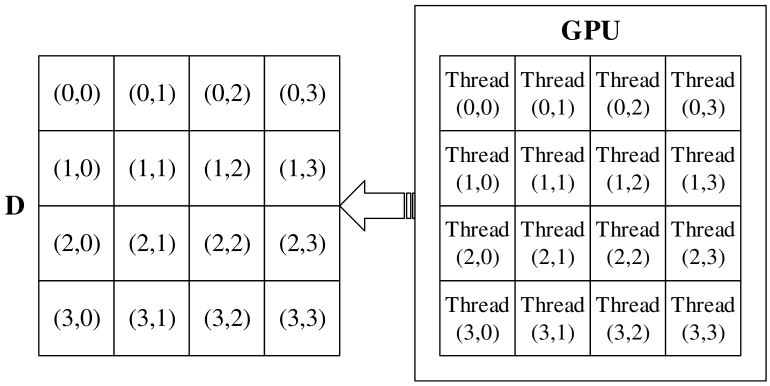

D
(0,0) (0,1) (0,2) (0,3)
(1,0) (1,1) (1,2) (1,3)
(2,0) (2,1) (2,2) (2,3)
(3,0) (3,1) (3,2) (3,3)
GPU
Thread (0,0) Thread (0,1) Thread (0,2) Thread (0,3)
Thread (1,0) Thread (1,1) Thread (1,2) Thread (1,3)
Thread (2,0) Thread (2,1) Thread (2,2) Thread (2,3)
Thread (3,0) Thread (3,1) Thread (3,2) Thread (3,3)

Fig. 2. Scheme of melody similarity calculation on GPU

5 Experiment and Evaluation

Several experiments were carried out to evaluate the performance of our GPU-accelerated music recommendation method. First, we introduce the evaluation environment. Then, we compare the accuracy between our method and the original EP algorithm executed on CPU. Finally, we test the performance of our method for large-scale library.

5.1 Experimental Environment

Experimental environment is shown in Table 2. We use an AMD AthlonII X4 3.2GHz for CPU, NVIDIA GTX480 with 480 CUDA cores for GPUs. For the GPU program, the number of threads per-block is 512 in GTX480.

Table 2. Experimental Environment

	CPU	GTX480
Number of Core	1~4	480 (15SM)
Processor Clock	3.2GHz	1.401GHz
Complier	GCC 4.1.2	CUDA SDK 4.0

We use three public music collections to evaluate our music recommendation method, as shown in Table 3. To make the evaluation reproducible all collections are standard collections which have already been used in the literature and are in most instances freely available for research.

Table 3. Comparison of three public music collections

Collections	**Songs**	**Genres**	**Length**
GTzan[24]	1000	10	30
ISMIR 2004[25]	1458	6	variable length
Homburg[26]	1886	9	10

In addition, in order to evaluate the performance of our music recommendation method for large-scale library, we also customize a collection containing 10000 Chinese pop music clips (each 30s length).

Definition: The nearest neighbor search accuracy *k-NN* is denoted as the average ratio of real similar songs in *k* recommended songs.

$$k\_NN = \frac{1}{m}\sum_{i=1}^{m}\frac{NN_{found}}{k} \tag{2}$$

5.2 Evaluation of Accuracy

In order to validate the accuracy of the proposed music recommendation method, we conduct a genre classification experiment on all public collections we have introduced in the previous section. The results are shown in Table III. Parameters: feature vector dimension $k = 40$, the filter parameter $fs = 5\%$.

As shown in Table 4, indexing and GPU-acceleration have no effect on the accuracy of EP algorithm.

When the music library increases, we also need to evaluate the accuracy of our music recommendation method. As shown in Fig.3, filter parameter *fs* has an important influence, which indicates that the bigger *fs*, the higher accuracy and computational complexity. Furthermore, as *fs* increasing, the feature vector dimension k has less impact on the accuracy.

Table 4. Comparison of genre classification in three public collections

Collection	Full scan	Indexing	Indexing+GPU
GTzan	72.5%	71.1%	71.1%
ISMIR2004	83.2%	84.5%	84.5%
Homburg	45.4%	44.8%	44.8%

When the music library increases, we also need to evaluate the accuracy of our music recommendation method. As shown in Fig.3, filter parameter *fs* has an important influence, which indicates that the bigger *fs*, the higher accuracy and computational complexity. Furthermore, as *fs* increasing, the feature vector dimension *k* has less impact on the accuracy.

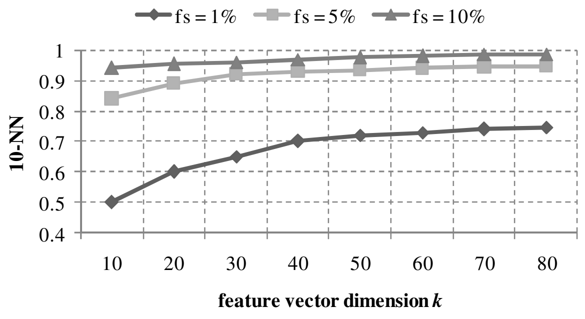


Fig. 3. 10-NN for different *k* and *fs*

Fig.4 indicates that the more nearest neighbors, the lower of *k-NN*. However, as *fs* increasing, the number of nearest neighbors has less impact on *k-NN*.

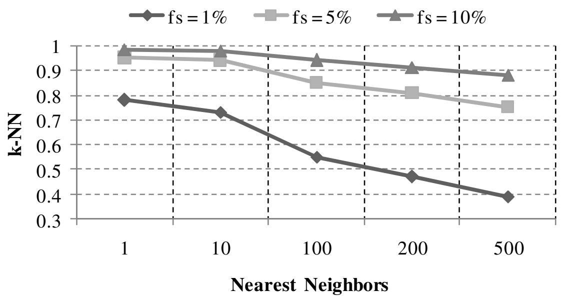


Fig. 4. *k-NN* for different *fs* and the nearest neighbors

5.3 Evaluation of Performance

The proposed music recommendation method combines indexing and GPU acceleration to expand the handled music library. As the copyright and other factors, we cannot establish a library similar to Amazon music store. Thus, based on the existing library, we evaluate the performance of our method by using specific performance indicators (such as query time).

Fig.5 indicates the performance of our proposed music recommendation method. We observe that query time is gradually reduced with the parameter *fs* reduction. Moreover, query time is further decreased when we accelerate the algorithms on GPU. For instance, if *fs* equals to 5% and the music library reaches 10000 clips, query time equals to 2.7 milliseconds. So, with the current acceleration, our proposed recommendation method can handle million magnitude libraries within 0.5 seconds and ten-million magnitude libraries within 3 seconds. Thus, our music recommendation method is capable to handle large-scale music library.

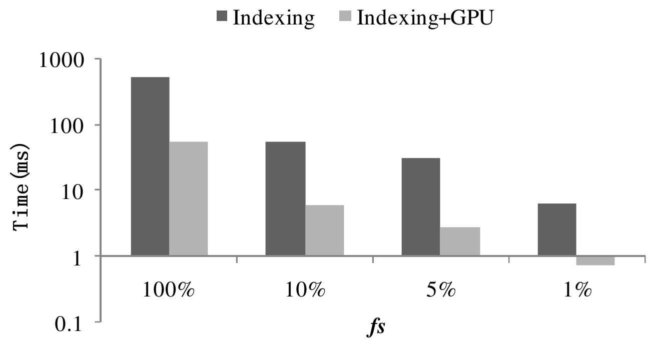


Fig. 5. Comparison of query time (k=40)

6 Conclusion and Future Works

This paper proposes a music recommendation method for large-scale music library, which combines indexing and GPU acceleration to improve the handled music library. Our experiments demonstrate that this method can handle the large-scale music library within the acceptable times. Meanwhile, we make full use of GPU's multithreading capabilities in the feature calculation, which shows that GPU has advantages in terms of the music signal.

For future works, we will further expand the scale of music library and optimize the GPU-based parallel program to improve the performance.

Acknowledgments. This research was funded by the National Natural Science Foundation of China under Grant No.61232009 and No. 61370059, the Doctoral Fund of Ministry of Education of China under Grant No.20101102110018, Beijing Natural Science Foundation under Grant No.4122042 and the fund of the State Key Laboratory of Software Development Environment under Grant No.SKLSDE-2014ZX-13.

References

1. Amazon MP3 Store, `http://www.amazon.com/MP3-MusicDownload/b/ref=topnav_storetab_dmusic?ie=UTF8&node=163856011`
2. Adomavicius, G., Tuzhilin, A.: Towards the Next Generation of Recommender Systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on knowledge and Data Engineering 17(6), 734–749 (2005)
3. Yianilos, P.: Data Structures and Algorithms for Nearest Neighbour Search in General Metric Spaces. In: ACM-SIAM International Conference on Discrete Algorithms, pp. 311–321 (1993)
4. Andoni, A., Indyk, P.: MIT, C.: Near-optimal Hashing Algorithms for Approximate Nearest Neighbour in High Dimensions. In: IEEE International Conference on Foundations of Computer Science, pp. 459–468 (2006)
5. Schnitzer, D., Flexer, A., Widmer, G.: A Fast Audio Similarity Retrieval Method for Millions of Music Tracks. Multimedia Tools Application (2010)
6. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: The Thirteenth Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
7. Casey, M., Slaney, M.: Song Intersection by Approximate Nearest Neighbor Search. In: 7th International Conference of Music Information Retrieval, pp. 144–149 (2006)
8. Cai, R., Zhang, C., Zhang, L., Ma, W.Y.: Scalable Music Recommendation by Search. In: The 15th International Conference on Multimedia, pp. 1065–1074 (2007)
9. Roy, P.: Aucouturie,r J. J., Pachet, F., Beurive, A.: Exploiting the Tradeoff Between Precision and Cpu-time to Speed Up Nearest Neighbor Search. In: 6th International Conference on Music Information Retrieval (2005)
10. Mandel, M., Ellis, D.P.: Labrosa's Audio Music Similarity and Classification Submissions. In: International Symposium on Music Information Retrieval (2007)

11. Levy, M., Sandler, M.: Lightweight Measures for Timbral Similarity of Musical Audio. In: the 1st ACM workshop on Audio and Music Computing Multimedia, pp. 27–36 (2006)
12. Yianilos, P.N.: Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In: The Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 311–321 (1993)
13. Athitsos, V., Potamias, M., Papapetrou, P., Kollios, G.: Nearest Neighbor Retrieval Using Distance-based Hashing. In: The IEEE 24th International Conference on Data Engineering, pp. 327–336 (2008)
14. Faloutsos, C., Lin, K.I.: Fastmap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In: ACM SIGMOD International Conference on Management of Data, pp. 163–174 (1995)
15. Cano, P., Kaltenbrunner, M., Gouyon, F., Batlle, E.: On the Use of Fastmap for Audio Retrieval and Browsing. In: The 3rd International Conference of Music Information Retrieval, pp. 275–276 (2002)
16. Gustavo, P., Alexander, L., Levada, M.: Voice Command Recognition with Dynamic Time Warping (DTW) using Graphics Processing Units (GPU) with CDUA. In: 19th International Symposium on Computer Architecture and High Performance Computing, pp. 19–25 (2007)
17. Ferraro, P., Hanna, P., Imbert, L., Izard, T.: Accelerating Query-By-Humming on GPU. In: 10th International Society for Music Information Retrieval Conference (2009)
18. Sanabria, A., Oyaga, J.V., Bonilla, C.P.: Fast Parallel Algorithm for Audio Content Retrieval on GPUs. In: the 6th International Conference on Colombian Computing Congress (2011)
19. MIREX HOME, `http://www.music-ir.org/mirex/wiki/MIREX_HOME`
20. Mandel, M., Ellis, D.: Song-level Features and Support Vector Machines for Music Classification. In: The 6th International Conference on Music Information Retrieval (2005)
21. Pampalk, E.: Computational Models of Music Similarity and Their Application in Music Information Retrieval. Doctoral Dissertation, Vienna University of Technology (2006)
22. Pohle, T.: Automatic Characterization of Music for Intuitive Retrieval. PhD thesis, Johannes Kepler University Linz (2010)
23. Yao, G.C., Zheng, Y., Xiao, L.M., Ruan, L., Li, Y.N.: Efficient Vocal Melody Extraction from Polyphonic Music Signals. Electronics and Electrical Engineering 19(6), 103–108 (2013)
24. GTzan, `http://marsyas.info/download/data_sets/`
25. ISMIR2004, `http://ismir2004.ismir.net/genre_contest/index.htm`
26. Homburg, `http://www-ai.cs.uni-dortmund.de/audio.html`
27. Xiao, L.M., Zheng, Y., Tang, W.Q., Yao, G.C., Ruan, L.: A GPU-Accelerated Large-Scale Music Similarity Retrieval Method. In: IEEE International Conference on Internet of Things (2013)
28. Xiao, L.M., Zheng, Y., Tang, W.Q., Yao, G.C., Ruan, L.: Parallelizing Dynamic Time Warping Algorithm Using Prefix Computations on GPU. In: 15th IEEE International Conference on High Performance Computing and Communications (2013)

GPU-Accelerated Verification of the Collatz Conjecture

Takumi Honda, Yasuaki Ito, and Koji Nakano

Department of Information Engineering, Hiroshima University,
Kagamiyama 1-4-1, Higashi Hiroshima 739-8527, Japan
{honda,yasuaki,nakano}@cs.hiroshima-u.ac.jp

Abstract. The main contribution of this paper is to present an implementation that performs the exhaustive search to verify the Collatz conjecture using a GPU. Consider the following operation on an arbitrary positive number: if the number is even, divide it by two, and if the number is odd, triple it and add one. The Collatz conjecture asserts that, starting from any positive number m, repeated iteration of the operations eventually produces the value 1. We have implemented it on NVIDIA GeForce GTX TITAN and evaluated the performance. The experimental results show that, our GPU implementation can verify 5.01×10^{11} 64-bit numbers per second, while the CPU implementation on Intel Xeon X7460 can verify 1.80×10^9 64-bit numbers per second. Thus, our implementation on the GPU attains a speed-up factor of 278 over the single CPU implementation.

Keywords: Collatz conjecture, GPGPU, Parallel processing, Exhaustive verification

1 Introduction

The Collatz conjecture is a well-known unsolved conjecture in mathematics [8,19,22]. Consider the following operation on an arbitrary positive number:

even operation if the number is even, divide it by two, and
odd operation if the number is odd, triple it and add one.

The Collatz conjecture asserts that, starting from any positive number, repeated iteration of the operations eventually produces the value 1. For example, starting from 3, we have the following sequence to produce 1.

$$3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

The exhaustive verification of the Collatz conjecture is to perform the repeated operations for numbers from 1 to the infinite as follows:

```
for $m \leftarrow 1$ to $\infty$ do
    begin
    $n \leftarrow m$
    while($n > 1$) do
        if $n$ is even then $n \leftarrow \frac{n}{2}$
        else $n \leftarrow 3n + 1$
end
```

Clearly, if the Collatz conjecture is not true, then the while-loop in the program above never terminates for a counter example m. A working project for the Collatz conjecture is currently checking 61-bit numbers [17].

There are several researches for accelerating the exhaustive verification of the Collatz conjecture. It is known [1,3,4,5] that series of even and odd operations for n can be done in one step by computing $n \leftarrow B[n_L] \cdot n_H + C[n_L]$ for appropriate tables B and C, where the concatenation of n_H and n_L corresponds to n. In [1,3,5], FPGA implementations have been presented to repeat the operations of the Collatz conjecture. These implementations perform the even and odd operations for some fixed size of bits of interim numbers. However, in [1], the implementation ignores the overflow. Hence, if there exists a counter example number m for the Collatz conjecture such that, infinitely large numbers are generated by the operations from m, their implementation may fail to detect it. On the other hand, in [3], the implementation can verify the conjecture for up to 23-bit numbers. This is not sufficient because a working project for the Collatz conjecture is currently checking 61-bit numbers [17]. In our previous paper [4], we have shown a software-hardware cooperative approach to verify the Collatz conjectures for 64-bit numbers n. This approach supports almost infinitely large interim numbers m. The idea is to perform the while-loop for interim values with up to 78 bits using a coprocessor embedded in an FPGA. If an interim value m has more than 78 bits, the original value n is reported to the host PC. The host PC performs the verification for such n using unlimited number of bits by software. This software-hardware cooperative approach makes sense, because

– the hardware implementation on the FPGA is fast and low power consumption, but the number of bits for the operation is fixed,
– the software implementation on the PC is relatively slow and high power consumption, but the number of bits for the operation is unlimited.

Additionally, in another previous paper of ours [5], we have proposed an efficient implementation of a coprocessor that performs the exhaustive search to verify the Collatz conjecture using embedded DSP slices on a Xilinx FPGA. By effective use of embedded DSP slices instead of multipliers used in [4], the coprocessor can perform the exhaustive verification faster than the above FPGA implementations.

The main contribution of this paper is to further accelerate the exhaustive verification for the Collatz conjecture using a GPU (Graphics Processing Unit). Recent GPUs can be utilized for general purpose parallel computation. We can use many processing units connected with an off-chip global memory in GPUs. CUDA (Compute Unified Device Architecture) [12] is an architecture for general purpose parallel computation on GPUs. Using CUDA, we can develop parallel processing programs to be implemented in GPUs. Therefore, many studies have been devoted to implement parallel algorithms using CUDA [2,6,7,9,10,16,18,20,21]. The ideas of our GPU implementation are (i) a GPU-CPU cooperative approach, (ii) efficient memory access for the global memory and the shared memory, and (iii) optimization of the code for arithmetic with larger integers. By effective use of a GPU and the above ideas, our new GPU implementation can verify 5.01×10^{11} 64-bit numbers per second. On the other hand, the CPU implementation can verify 1.80×10^{9} 64-bit numbers per second. As far as we know, the FPGA implementation in [5] has been the fastest implementation. However, our GPU implementation can verify the Collatz conjecture 3.05 times faster the FPGA implementation.

This paper is organized as follows. Section 2 presents several techniques for accelerating the verification of the Collatz conjecture. In Section 3, we show the GPU and CUDA architectures to understand our idea. Section 4 proposes our new ideas to implement the verification of the Collatz conjecture on the GPU. The experimental results are shown in Section 5. Finally, Section 6 offers concluding remarks.

2 Accelerating the Verification of the Collatz Conjecture

The main purpose of this section is to introduce an algorithm for accelerating the verification of the Collatz conjecture. The basic ideas of acceleration are shown in [8,22].

The first technique is to terminate the operations before the iteration produces 1. Suppose that we have already verified that the Collatz conjecture is true for numbers less than n, and we are now in position to verify it for number n. Clearly, if we repeatedly execute the operations for n until the value is 1, then we can confirm that the conjecture is true for n. Instead, if the value becomes n' for some n' less than n, then we can guarantee that the conjecture is true for n because it has been proved to be true for n'. Thus, it is not necessary to repeat this operation until the value is 1, and we can terminate the iteration when, for the first time, the value is less than n.

The second technique is to perform several operations in one step. Consider that we want to perform the operations for n and let n_L and n_H be the least significant two bits and the remaining bits of n. In other words, $n = 4n_H + n_L$ holds. Clearly, the value of n_L is either 00, 01, 10, or 11. We can perform the several operations for n based on n_L as follows:

$n_L = 00$: Since two even operations are applied, the resulting number is n_H.

$n_L = 01$: First, odd operation is applied and the resulting number is $(4n_H + 1) \cdot 3 + 1 = 12n_H + 4$. After that, two even operations are applied, and we have $3n_H + 1$.

$n_L = 10$: First, even operation is performed and we have $2n_H + 1$. Second, odd operation is applied and we have $(2n_H + 1) \cdot 3 + 1 = 6n_H + 4$. Finally, by even operation, the value is $3n_H + 2$.

$n_L = 11$: First, odd operation is applied and we have $(4n_H+3)\cdot 3+1 = 12n_H+10$. Second, by even operation, the value is $6n_H + 5$. Again, odd operation is performed and we have $(6n_H + 5) \cdot 3 + 1 = 18n_H + 16$. Finally, by even operation, we have $9n_H + 8$.

For example, if $n_L = 11$ then we can obtain $9n_H + 8$ by applying 4 operations, odd, even, odd, and even operations in turn. Let B and C be tables as follows:

	B	C
00	1	0
01	3	1
10	3	2
11	9	8

Using these tables, we can perform the following table operation, which emulates several odd and even operations:

table operation For least significant two bits n_L and the remaining most significant bits n_H of the value, the new value is $B[n_L] \cdot n_H + C[n_L]$.

Let us extend the table operation for least significant two bits to d bits. For an integer $n \geq 2^d$, let n_L and n_H be the least significant d bits, that is, $n = 2^d n_H + n_L$. We call d is *the base bits.* Suppose that, the even or odd operations are repeatedly performed on $n = 2^d n_H + n_L$. We use two integers a and b such that $n = b \cdot n_H + c$ to denote the current value of n. Initially, $b = 2^d$ and $c = n_L$. We repeatedly perform the following rules for b and c.

even rule If both b and c are even, then divide them by two.
odd rule If c is odd, then triple b, and triple c and add one.

These two rules are applied until no more rules can be applied, that is, until b is odd. It should be clear that, even and odd rules correspond to even and odd operations of the Collatz conjecture. If i even rules and j odd rules applied, then the value of b is $2^{d-i}3^j$. Thus, exactly d even rules are applied until the termination. After the termination, we can determine the value of elements in tables B and C such that $B[n_L] = b$ and $C[n_L] = c$. Using tables B and C, we can perform the table operation for d bits n_L, which involves d even operations and zero or more odd operations. In this way, we can accelerate the operation of the Collatz conjecture. In paper [1], we have implemented for various numbers of bits of n_L. Our GPU implementation results show that the performance is well balanced when the number of bits of n_L is 11.

The third technique to accelerate the verification of the Collatz conjecture is to skip numbers n such that we can guarantee that the resulting number is less than n after the table operation. For example, suppose we are using two bit table and $n_H > 0$. If $n_L = 00$ then the resulting value is n_H, which is less than n. Thus, we can skip the table operation for n if $n_L = 00$. If $n_L = 01$ then the resulting value is $3n_H + 1$, which is always less than $n = 4n_H + 1$, and we can skip the table operation. Similarly, if $n_L = 10$ then we can skip the table operation. On the other hand $n_L = 11$ then the resulting value is $9n_H + 8$, which is always larger than n. Therefore, the Collatz conjecture is guaranteed to be true whenever $n_L \neq 11$, because it has been verified true for numbers less than n. Consequently, we need to execute the table operation for number n such that $n_L = 11$.

We can extend this idea for general case. For least significant d bits n_L, we say that n_L is not *mandatory* if the value of b is less than 2^d at some moment while even and odd rules are repeatedly applied. We can skip the verification for non-mandatory n_L. The reason is as follows: Consider that for number n, we are applying even and odd rules. Initially, $b = 2^d$ and $c \leq 2^d - 1$ hold. Thus, while even and odd rules are applied, $b > c$ always hold. Suppose that $b \leq 2^d - 1$ holds at some moment while the rules are applied. Then, the current value of n is

$$bn_H + c < bn_H + b \leq (2^d - 1)n_H + b < 2^d n_H \leq n.$$

It follows that, the value is less than n when the corresponding even and odd operations are applied. Therefore, we can omit the verification for numbers that have no mandatory least significant bits.

For least significant d bit number, we use table S to store the mandatory least significant bits. Let s_d be the number of such mandatory least significant bits. Using these tables, we can write a verification algorithm as follows:

```
**for** $m_H \leftarrow 1$ **to** $+\infty$ **do**
    **for** $i \leftarrow 0$ **to** $s_d - 1$ **do**
        **begin**
            $m_L \leftarrow S[i]$;
            $n \leftarrow m \leftarrow 2^d m_H + m_L$;
            **while**$(n \geq m)$ **do**
                **begin**
                    Let $n_L$ be the least significant $d$ bits and
                        $n_H$ be the remaining bits.
                    $n \leftarrow B[n_L] \cdot n_H + C[n_L]$;
                **end**
        **end**
```

For the benefit of readers, we show B, C, and S for 4 base bits in Table 1. From $s_4 = 3$, we have 3 mandatory least significant bits out of 16.

For the reader's benefit, Table 2 shows the necessary word size for each of tables B and C for each base bit. It also shows the expected number of odd/even operations included in one step operation $n \leftarrow B[n_L] \cdot n_H + C[n_L]$. Table 3 shows the size of table S. It further shows the ratio of the mandatory numbers over all numbers. Later, we set base bit 11 for tables B and C, and base bit 32 for table S in our proposed GPU implementation. Thus, one operation $n \leftarrow B[n_L] \cdot n_H + C[n_L]$ corresponds to expected 16.5 odd/even operations. Also, we skip approximately 99.04% of non-mandatory numbers.

Table 1. Tables B, C, and S for least significant 4 bits

	B	C	S
0000	1	0	0111
0001	9	1	1011
0010	9	2	1111
0011	9	2	-
0100	3	1	-
0101	3	1	-
0110	9	4	-
0111	27	13	-
1000	3	2	-
1001	27	17	-
1010	3	2	-
1011	27	20	-
1100	9	8	-
1101	9	8	-
1110	27	26	-
1111	81	80	-

Table 2. The size of tables B and C

base bit	words	operation
4	16	6.0
5	32	7.5
6	64	9.0
7	128	10.5
8	256	12.0
9	512	13.5
10	1k	15.0
11	2k	16.5
12	4k	18.0
13	8k	19.5
14	16k	21.0
15	32k	22.5
16	64k	24.0

Table 3. The size of tables S

base bit	words	ratio	base bit	words	ratio
3	2	0.2500	18	7495	0.0286
4	3	0.1875	19	14990	0.0286
5	4	0.1250	20	27328	0.0261
6	8	0.1250	21	46611	0.0222
7	13	0.1016	22	93222	0.0222
8	19	0.0742	23	168807	0.0201
9	38	0.0742	24	286581	0.0171
10	64	0.0625	25	573162	0.0171
11	128	0.0625	26	1037374	0.0155
12	226	0.0552	27	1762293	0.0131
13	367	0.0448	28	3524586	0.0131
14	734	0.0448	29	6385637	0.0119
15	1295	0.0395	30	12771274	0.0119
16	2114	0.0323	31	23642078	0.0110
17	4228	0.0323	32	41347483	0.0096

3 GPU and CUDA Architectures

Figure 1 illustrates the CUDA hardware architecture. CUDA uses three types of memories in the NVIDIA GPUs: *the global memory*, *the shared memory*, and *the registers* [14]. The global memory is implemented as an off-chip DRAM of the GPU, and has large capacity, say, 1.5-6 Gbytes, but its access latency is very long. The shared memory is an extremely fast on-chip memory with lower capacity, say, 16-48 Kbytes. The registers in CUDA are placed on each core in the multiprocessor and the fastest memory, that is, no latency is necessary. However, the size of the registers is the smallest during them. The efficient usage of the global memory and the shared memory is a key for CUDA developers to accelerate applications using GPUs. In particular, we need to consider *the coalescing* of the global memory access and *the bank conflict* of the shared memory access [11,13]. To maximize the bandwidth between the GPU and the DRAM chips, the consecutive addresses of the global memory must be accessed in the same time. Thus, threads should perform coalescing access when they access to the global memory. Also, CUDA supports broadcast access to the shared memory without the bank conflict [14]. The broadcast access is a shared memory request such that two or more threads refer the same address. Thus, in our GPU implementation, to make memory access efficient, we perform the coalescing and the broadcast access for the reference to tables B and C stored in the global memory and the shared memory as possible, respectively.

CUDA parallel programming model has a hierarchy of thread groups called *grid*, *block* and *thread*. A single grid is organized by multiple blocks, each of which has equal number of threads. The blocks are allocated to streaming processors such that all threads in a block are executed by the same streaming processor in parallel. All threads can access to the global memory. However, as we can see in Figure 1, threads in a block can access to the shared memory of the

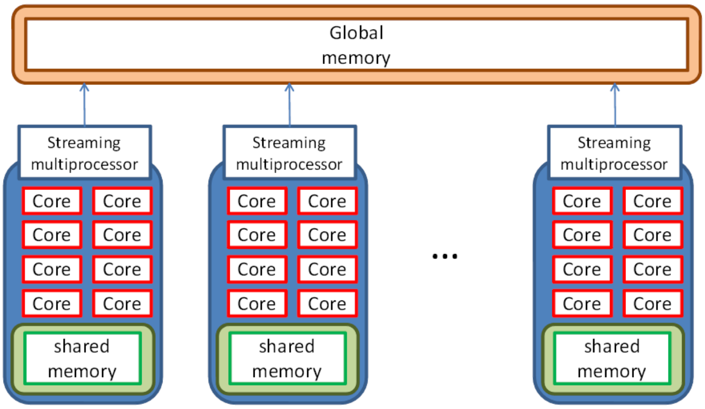


Fig. 1. CUDA hardware architecture

streaming processor to which the block is allocated. Since blocks are arranged to multiple streaming processors, threads in different blocks cannot share data in shared memories. Also, the registers are only accessible by a thread, that is, the registers cannot be shared by multiple threads.

CUDA C extends C language by allowing the programmer to define C functions, called *kernels*. By invoking a kernel, all blocks in the grid are allocated in streaming processors, and threads in each block are executed by processor cores in a single streaming processor. In the execution, threads in a block are split into groups of thread called *warps*. Each of these warps contains the same number of threads and is executed independently. When a warp is selected for execution, all threads execute the same instruction. When one warp is paused or stalled, other warps can be executed to hide latencies and keep the hardware busy.

There is a metric, called *occupancy*, related to the number of active warps on a streaming processor. The occupancy is the ratio of the number of active warps per streaming processor to the maximum number of possible active warps. It is important in determining how effectively the hardware is kept busy. The occupancy depends on the number of registers, the numbers of threads and blocks, and the size of shard memory used in a block. Namely, utilizing too many resources per thread or block may limit the occupancy. To obtain good performance with the GPUs, the occupancy should be considered.

4 GPU Implementation

The main purpose of this section is to show a GPU implementation of verifying the Collatz conjecture. The ideas of our GPU implementation are

(i) a GPU-CPU cooperative approach,
(ii) efficient memory access for the global memory and the shared memory, and
(iii) optimization of the code for arithmetic with larger integers.

The details of our GPU implementation using these ideas are described, as follows.

4.1 A GPU-CPU Cooperative Approach

In the following, we show a GPU-CPU cooperative approach that is similar to the idea of a hardware-software cooperative approach in [4]. In this paper, we assume that 64-bit numbers are verified. This assumption is sufficient because a working project for the Collatz conjecture is currently checking 61-bit numbers [17]. We note that the verified numbers can be extended easily since the interim numbers in the verification can be larger than 64-bit numbers. In the verification of the Collatz conjecture, therefore, arithmetic with larger integers having more than 64 bits is necessary to compute $B[n_L] \cdot n_H + C[n_L]$. Depending on an initial value, the size of the interim value may become very large during the verification. If larger interim value is allowed in the computation on the GPU, the values cannot be stored on the registers, that is, they have to be stored on the global memory whose access latency is very long. In our implementation, therefore, the maximum size of interim values is limited to 96 bits, which consists of three 32-bit integers, to perform the computation only on the registers. By limiting the maximum size, the computation can be performed as fixed length computation without overhead caused by arbitrary length computation. Suppose that a thread finds that the interim value is overflow for the initial value m. The thread reports m through the global memory. After all the threads finish the verification, the host program checks whether there are overflows or not. If overflows are found, the host verifies the Collatz conjecture for the values using unlimited number of bits by software on the CPU.

The reader may think that if the number of overflows is larger, the verification time is longer. However, the number of overflows is small enough for the limitation of 96 bits [5]. Therefore, it is reasonable to perform the verification for overflow numbers on the host. In Section 5, we will evaluate the number of overflows and the verification time for them.

4.2 Efficient Memory Access for the Global Memory and the Shared Memory

In order to reduce the global memory access, all the contents of tables B and C stored in the global memory are cached on the shared memory. Threads in each block load the contents of the tables B and C to the shared memory at first. In this case, threads read them from the global memory with the coalescing access. After that, each thread verifies assigned numbers. In our implementation, we use tables B and C with base bit 11. Each entry of tables B and C is stored as a 32-bit integer. According to Table 2, the number of words of tables B and C is 2048 and the total size of these tables is 16 Kbytes. Since the maximum size of the shared memory is 48 Kbytes, tables B and C with base bit 12 can be stored on the shared memory. However, since for that case, the size of utilized shared memory is too much, the occupancy is decreased, that is, the performance becomes lower. Thus, we use tables B and C with base bit 11 and cache them on the shared memory.

In addition, we consider efficient memory access of cached tables B and C on the shared memory. Recall that the address of the reference for tables B and C is always determined by the value of n_L which is the least significant bits of interim value n. If interim values have distinct least significant bits, the access for tables B and C becomes distinct. This occurs the bank conflict of the shared memory. To reduce this bank conflict, we utilize the broadcast access, that does not occur the bank conflict. In our GPU implementation, we arrange initial values verified by threads in a block such that the least significant bits of them are identical. More specifically, the data format of them is shown in Fig. 2. In the figure, *thread_ID* denotes a thread index within a block, *block_ID* denotes a block index within a kernel, and M is a constant. In each block, $S[block\_ID]$ and M are common values for threads and each thread in a block verifies the Collatz conjecture for $2^8(= 256)$ initial values. Using this arrangement, threads in a block concurrently verify the conjecture for values that are identical except *thread_ID*. Since the values are not exactly identical, we cannot avoid the bank conflict for all the access. However, until the bits depending on the *thread_ID* are included into n_L, threads in a block can refer the identical address of tables B and C at the same time. For each iteration of the while-loop in the algorithm in Section 2, the interim value is divided into the least significant d bits and the remaining bits, that is, the value is d-bit-right-shifted. Therefore, using the data format in Fig. 2, threads can refer the same address $\lfloor \frac{8+(45-b)+b}{d} \rfloor = \lfloor \frac{53}{d} \rfloor$ times for each verification. For example, when $d = 11$, that is the optimal parameter in our experiment, threads can refer the same address at least 4 times for each initial value.

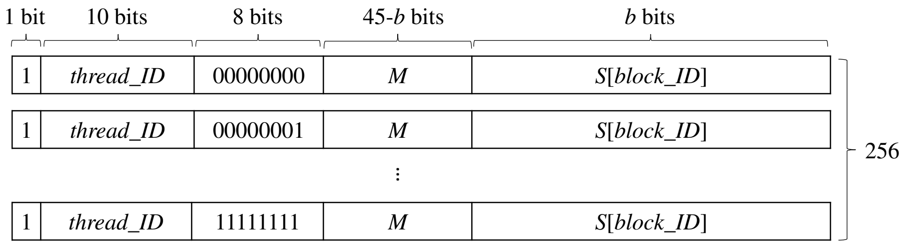


Fig. 2. The data format of 64-bit numbers verified by each thread in a block, where *thread_ID* denotes a thread index within a block, *block_ID* denotes a block index within a kernel, and M is a constant

4.3 Optimization of the Code for Arithmetic with Larger Integers

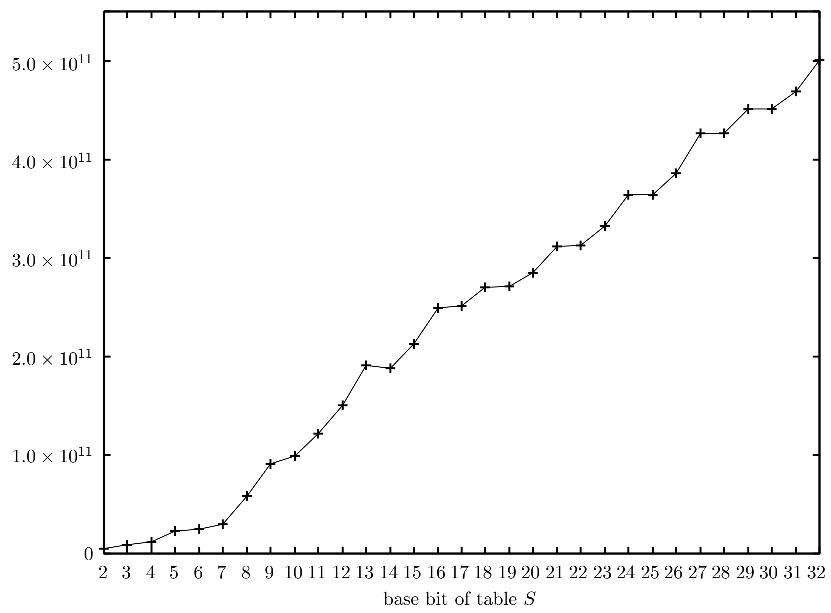
As mentioned in the above, arithmetic with larger integers having more than 64 bits is necessary to compute $B[n_L] \cdot n_H + C[n_L]$. In C language, however, there is no efficient way of doing such arithmetic because C language does not support operations with the carry flag bit. In a common way to perform the arithmetic with larger integers, 32-bit operations are performed on 64-bit operations by extending the bit-length. However, the overhead of type conversion for the extension of the bit-length cannot be ignored. To optimize the arithmetic with larger integers, therefore, a part of the code is written in PTX [15] that is an assembly language for NVIDIA GPUs and can be used as inline assembler in CUDA C language. PTX supports arithmetic operations with the carry flag bit. Concretely, we use *mad* and *madc* that are 32-bit arithmetic operations in PTX to compute $B[n_L]\cdot n_H+C[n_L]$. These operations multiply two 32-bit integers and add one 32-bit integer excluding and including the carry flag bit, respectively. Applying the optimization of the code, in the preliminary experiment, the result shows that the optimized implementation can verify the Collatz conjecture approximately 1.8 times faster than the non-optimized implementation.

5 Performance Evaluation

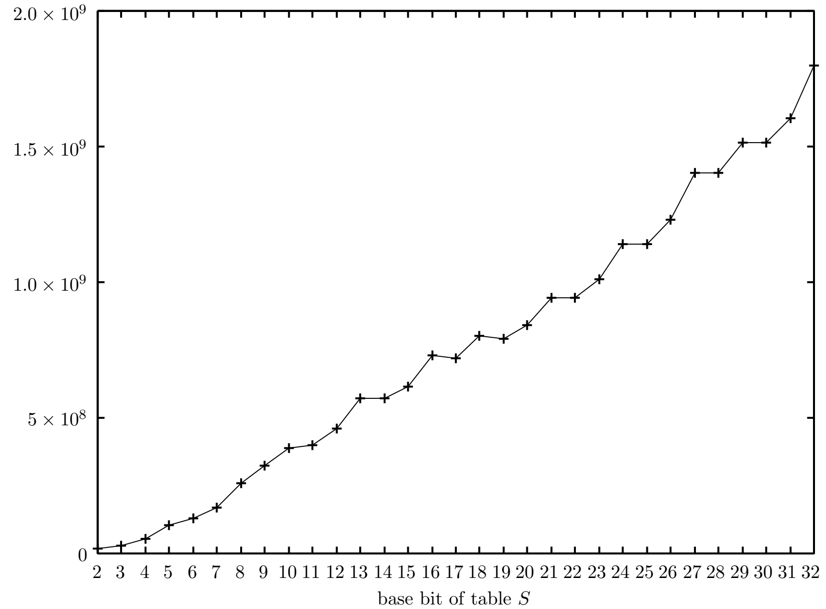
We have implemented our GPU implementation of verifying the Collatz conjecture using CUDA C. We have used NVIDIA GeForce GTX TITAN with 2688 processing cores (14 Streaming Multicore processors which have 192 processing cores each) running in 876 MHz and 6 GB memory. For the purpose of estimating the speed up of our GPU implementation, we have also implemented a software approach of verifying the Collatz conjecture using GNU C. In the software implementation, we can apply the idea of accelerating the verification described in Section 2. The difference from the GPU implementation is that the software implementation uses unlimited number operations and verifies the Collatz conjecture serially. Each of them will be compared in the following. We have used in Intel Xeon X7460 running in 2.66GHz and 128GB memory to run the CPU implementation.

We have evaluated the computing time of the GPU implementation by verifying the Collatz conjecture for the 64-bit numbers whose data format is shown in Fig. 2. For this purpose, we have 10 randomly generated integers as a constant M. For each generated integer M, the GPU implementation verified the Collatz conjecture. In the GPU and CPU implementation, we use tables B and C with base bit 11 and 12, which are optimal bits obtained by our experiments, respectively.

Fig. 3 shows the number of verified numbers per second for various base bit of table S in the GPU and CPU implementations. We note that in the GPU implementation, the computing time of the verification for overflow numbers by the CPU is included as described in Section 4. For example, when base bit of table S is 32 for a constant M in the GPU verification, 29764 overflow numbers were found, that is, the size of interim values for 29764 numbers became more than 96 bits. After that, the host program verified the conjecture for these numbers using unlimited number of bits by software. The verification time in the CPU was 65 *ms* including the time of data transfer between the GPU and CPU. Since the



(a) GPU



(b) CPU

Fig. 3. The number of verified 64-bit numbers per second for various size of base bit of table S

total computing time was 2249144 ms, the verification time for overflow numbers by the CPU is much shorter. According to the both graphs, when the base bit is larger, the number is larger because the number of non-mandatory numbers is larger for larger base bit as shown in Table 3. For table S with base bit 32, our GPU implementation can verify the Collatz conjecture for 5.01×10^{11} numbers per second. On the other hand, in the CPU implementation, for table S with base bit 32, the CPU implementation can verify the Collatz conjecture for 1.80×10^9 numbers per second. Thus, our GPU implementation attains a speed-up factor of 278 over the CPU implementation. As far as we know, the FPGA implementation in [5] has been the fastest implementation. However, our GPU implementation can verify the Collatz conjecture 3.05 times faster the FPGA implementation.

6 Conclusions

We have presented a GPU implementation that performs the exhaustive search to verify the Collatz conjecture. In our GPU implementation, we have considered programming issues of the GPU architecture such as the coalescing of the global memory, the shared memory bank conflict, and the occupancy of the multicore processors. We have implemented it on NVIDIA GeForce GTX TITAN. The experimental results show that it can verify 5.01×10^{11} 64-bit numbers per second. On the other hand, the CPU implementation verifies 1.80×10^9 64-bit numbers. Thus, our GPU implementation attains a speed-up factor of 278.

References

1. An, F., Nakano, K.: An architecture for verifying Collatz conjecture using an FPGA. In: Proc. of the International Conference on Applications and Principles of Information Science, pp. 375–378 (2009)
2. Diaz, J., Muñoz-Caro, C., Niño, A.: A survey of parallel programming models and tools in the multi and many-core era. IEEE Transactions on Parallel and Distributed Systems 23(8), 1369–1386 (2012)
3. Ichikawa, S., Kobayashi, N.: Preliminary study of custom computing hardware for the 3x+1 problem. In: Proc. of IEEE TENCON 2004, pp. 387–390 (2004)
4. Ito, Y., Nakano, K.: A hardware-software cooperative approach for the exhaustive verification of the Collatz conjecture. In: Proc. of International Symposium on Parallel and Distributed Processing with Applications, pp. 63–70 (2009)
5. Ito, Y., Nakano, K.: Efficient exhaustive verification of the Collatz conjecture using DSP blocks of Xilinx FPGAs. International Journal of Networking and Computing 1(1), 49–62 (2011)
6. Ito, Y., Nakano, K.: A GPU implementation of dynamic programming for the optimal polygon triangulation. IEICE Transactions on Information and Systems E96-D(12), 2596–2603 (2013)
7. Ito, Y., Ogawa, K., Nakano, K.: Fast ellipse detection algorithm using Hough transform on the GPU. In: Proc. of International Conference on Networking and Computing, pp. 313–319 (December 2011)

8. Lagarias, J.C.: The 3x+1 problem and its generalizations. The American Mathematical Monthly 92(1), 3–23 (1985)
9. Man, D., Nakano, K., Ito, Y.: The approximate string matching on the hierarchical memory machine, with performance evaluation. In: Proc. of the IEEE 7th International Symposium on Embedded Multicore SoCs, pp. 79–94 (2013)
10. Man, D., Uda, K., Ito, Y., Nakano, K.: Accelerating computation of Euclidean distance map using the GPU with efficient memory access. International Journal of Parallel, Emergent and Distributed Systems 28(5), 383–406 (2013)
11. Man, D., Uda, K., Ueyama, H., Ito, Y., Nakano, K.: Implementations of a parallel algorithm for computing Euclidean distance map in multicore processors and GPUs. International Journal of Networking and Computing 1(2), 260–276 (2011)
12. NVIDIA Corp.: CUDA ZONE, https://developer.nvidia.com/cuda-zone
13. NVIDIA Corp.: CUDA C Best Practice Guide Version 5.5 (2013)
14. NVIDIA Corp.: CUDA C Programming Guide Version 5.5 (2013)
15. NVIDIA Corp.: Parallel Thread Execution ISA Version 3.2 (2013)
16. Ogawa, K., Ito, Y., Nakano, K.: Efficient Canny edge detection using a GPU. In: International Workshop on Advances in Networking and Computing, pp. 279–280 (November 2010)
17. Roosendaal, E.: On the 3x + 1 problem, http://www.ericr.nl/wondrous/index.html
18. Ryoo, S., Rodrigues, C.I., Baghsorkhi, S.S., Stone, S.S., Kirk, D.B., Hwu, W.-M.W.: Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In: Proc. of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 73–82 (2008)
19. Silva, T.O.: Maximum excursion and stopping time record-holders for the $3x + 1$ problem: Computational results. Mathematics of Computation 68(225), 371–384 (1999)
20. Takeuchi, Y., Takafuji, D., Ito, Y., Nakano, K.: ASCII art generation using the local exhaustive search on the GPU. In: Proc. of International Symposium on Computing and Networking, pp. 194–200 (2013)
21. Uchida, A., Ito, Y., Nakano, K.: Accelerating ant colony optimisation for the travelling salesman problem on the GPU. International Journal of Parallel, Emergent and Distributed Systems 29(4), 401–420 (2014)
22. Weisstein, E.W.: Collatz problem. From MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/CollatzProblem.html

Reducing the Interconnection Length for 3D Fault-Tolerant Processor Arrays

Guiyuan Jiang[1], Jigang Wu[2,⋆], Jizhou Sun[1], and Longting Zhu[2]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China 300072

[2] School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin, China 300387

{jguiyuan,asjgwu}@gmail.com

Abstract. The three-dimensional (3D) processor array has benefits of reducing interconnection latency, consuming less power and improving bandwidths compared to 2D processor arrays. However, it suffers from frequent faults due to power overheating during massively parallel computing. To achieve fault-tolerance under such a such a scenario, an effective method is to construct a non-faulty sub-array from the faulty array as large as possible, such that the original application can still work on the sub-array. However, logical sub-arrays produced by previous works contain large number of long interconnects, which leads to more communication cost, capacitance and dynamic power dissipation. In this paper, we investigate the problem of reducing the interconnection length of a logical array. First, we prove that it is a NP-hard problem. Then we propose an efficient heuristic to reduce the interconnection redundancy of a logical array by reducing the number of long interconnects in each logical plane. Each logical plane is optimized based on statistical information. Experimental results show that, on $32 \times 32 \times 32$ host array with fault densities ranging from 0.1% to 5%, the proposed algorithm is capable of reducing the interconnection length by 49.7% and 29.8% in average compared to the existing algorithm `GPR` and `CAR`, respectively.

Keywords: 3D processor array, fault tolerance, interconnection length, algorithm.

1 Introduction

The quest for high-performance and low-power leads to design multi-core architectures where an increasing number of processing elements (PEs) are integrated on a single chip in a tightly coupled fashion. Three dimensional (3D) processor arrays have been well-known for lower interconnection latency and larger bandwidth compared to 2D processor arrays. However, with the increased density, faults occur frequently due to power overheating during the massively parallel computing.

⋆ Corresponding author.

Two types of fault tolerance methods, i.e., *microarchitecture-level* approach and *PE-level* approach, are commonly investigated. In the first type, fault tolerant circuits are incorporated into each PE to achieve fault tolerance[1,2]. However, with the ever-increasing circuit density, obtaining high reliability of each PE is increasingly difficult and will become unaffordable in the near future. In PE-level approach, fault tolerance is achieved by reconfiguring interconnections among PEs to construct a fault-free sub-array. In other words, the faulty PEs are replaced or bypassed and the remaining fault-free PEs form a logical array. In fact, some many-core systems have employed the PE-level fault-tolerance by replacing faulty PEs with spare on-chip PEs, e.g., Sun's UltraSPARC T1 processor [3] and Azul's Vega2 chip [4]. It is reported in [5] that PE-level fault tolerance outperforms microarchitecture-level fault tolerance in term of yield-adjusted throughput and relatively smaller instructions per cycle with the fabrication technology advancing to 100 *nm*.

In PE-level fault tolerance, it is important that a fault-free sub-array must maintain the isomorphism with the original array. It requires the network of this sub-array is regular. If the regular network is not provided, programmers need to optimize the application according to the irregular network [6] or customize the network to match the traffic pattern of the application [7,8,9], in order to fully exploit the capabilities of processing. This is a big burden for programmers because an optimized program for one network may not work well for a different one and the programmers are facing various irregular networks as the faults occur randomly and cannot be predicted.

Reported approaches on the switch-based architecture can be classified into two categories, i.e., *the redundancy approach* [10,11] and *the degradation approach*[12,13,14]. The redundancy approach intends to obtain a fault-free logical array with guaranteed size, while the latter tries to provide a target array as large as possible. For 3D processor arrays[16][17], few works have been reported. To the best of our knowledge, [18] present the first work for 3D degradable processor arrays. Firstly, it propose an algorithm named `GPR` to construct a logical array as large as possible. To do that, `GPR` produces as many as possible logical 'planes' and then connects these planes to form a 3D logical array. After that, [18] proposes an algorithm named `CAR` to reduce the interconnection length of the logical array produced by `GPR`. To do that, `CAR` revises each logical plane to reduce the number of long interconnects (*nlis*).

Even algorithm `CAR` tried to optimize the interconnection length, the logical array by `CAR` still contains large *nlis*. It is because, in revising each logical plane, the position information of only 2 out of each PEs' 4 logical neighbors is used to to reduce *nlis*, which results in significant interconnection redundancy. On the other hand, algorithm `CAR` is time consuming (running in exponential time in the worst case), as it works in backtrack manners. Motivated by this, we develop an efficient algorithm to further reduce the *nlis*. In our approach, global statistical information is calculated and utilized to assist the optimization, also position information of all 4 logical neighbours of each PEs is used in revising each logical plane. Moreover, we accelerate the procedure of revising each plane by identifying and eliminating backtrack steps, so that our algorithm runs in nearly linear time.

The rest of the paper is organized as follows. Section II introduces some preliminaries regarding to important notations and the fault-tolerant architecture. In section III, we proving that reducing the interconnection length of a logical array is NP-hard problem. In section IV, we present the proposed heuristic which which optimize the interconnection length by revising each logical plane. In section V, we provide experimental results with analysis, and conclude our work in section V.

2 Preliminaries

2.1 Fault-Tolerant Architecture and Reconfiguration Schemes

Let H denote a physical/logical array containing $m \times n \times h$ processing elements (PEs), where m, n and h are the sizes of its x-axis dimension, y-axis dimension and z-axis dimension, respectively. Some of the PEs in H are faulty. Assume the fault density of the physical array is ρ $(0<\rho<1)$, then there are $N = (1-\rho)\cdot m \times n \times h$ fault-free PEs in H. A degradable subarray containing no faulty PEs can be constructed by changing the connections among PEs. The degradable subarray is called a target array or logical array, denoted as T. The planes in physical/logical array are called physical/logical planes, respectively. In this paper, $e_{x,y,z}$ $(e'_{x,y,z})$ indicates the PE located at (x, y, z) of the host (logical) array. Let $X(u)$, $Y(u)$, $Z(u)$ denote the x-axis index, y-axis index, z-axis index of PE u. $u = v$ indicates that PE u is identical to v. Let $H_{i,j}$ be a subset of H that each PE in $H_{i,j}$ is with x-axis index i and y-axis index j, i.e., $H_{i,j} = \{e_{i,j,k} \mid 1 \le k \le h\}$. Therefore, $H = \bigcup_{1\le i\le m, 1\le j\le n} H_{i,j}$.

Fig. 1. shows a host array with size of 3×3×2. The gray shaded boxes in the figures represent faulty PEs, while other boxes represent the fault-free PEs. Each circle represents a configuration switch, and there are 15 states for each switch [16][17], as shown in Fig. 1. Each link of the network is of capacity 1. The fault-tolerant reconfiguration is achieved by inserting several switches in the network

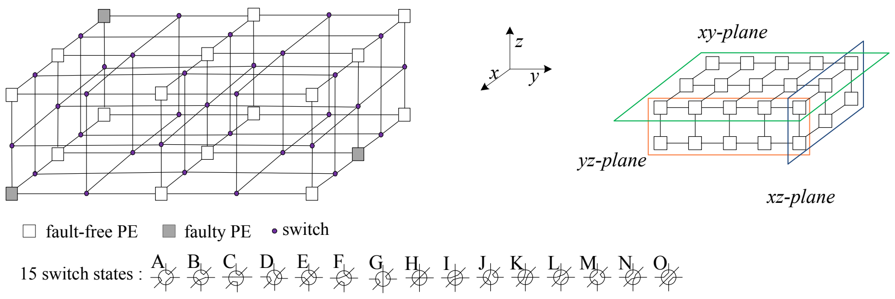
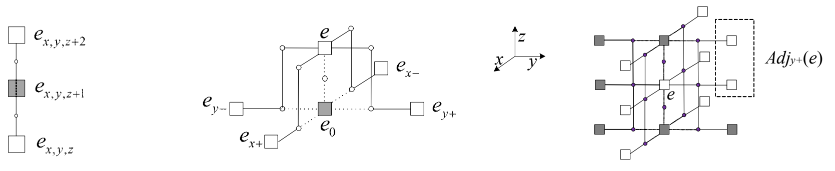


Fig. 1. Fault tolerance architecture and reconfiguration schemes



(a) z-direction bypass (b) xy-plane rerouting for replacing e_0 with e (c) e is unconnectable due to $|Adj_{y-}(e)|=0$

Fig. 2. Reconfiguration schemes to construct logical arrays by bypassing or replacing faulty PEs

allowing the network to dynamically change connections among PEs. In this switch based processor array, once the connection is set up, signal messages can be transferred through the connection without any header information. The time delay is negligible since no routing or arbitration is needed.

If PE $e_{x,y,z+1}$ faults as shown in Fig. 2 (a), then $e_{x,y,z}$ can communicate with $e_{x,y,z+2}$ directly and data will bypass $e_{x,y,z+1}$ without being processed, through an internal bypass links. This scheme is called *z-direction bypass scheme* (z-bypass for short). Under this scheme, an entire xy-plane of PEs can be bypassed if the plane contains too many faulty PEs. Note that no external switch is needed in this technique. For the layout of PEs, switches, links, I/O port selector, bypass controller and some practical issues related to the reconfigurable arrays, we refer the reader to [19]. In a similar manner, we can obtain the x-bypass and y-bypass scheme.

The *xy-plane rerouting scheme* defines rules to form a logical xy-plane. As shown in Fig. 2 (b), to form a fault-free logical plane, the faulty PE e_0 can be replaced with e by connecting e to e_{x-}, e_{x+}, e_{y-} and e_{y+}, respectively. In *xy-plane rerouting scheme*, PE e can directly connect to a logical neighbor if their z-axis distance does not exceed a fixed value d, by changing the states of switches. Formally, PE e can connect to a logical neighbor e_{y+} lying in e's $y+$ direction, if $|Z(e) - Z(e_{y+})| \leq d$ where d is called compensation distance.. The xy-plane rerouting scheme allows PEs from different physical xy-planes to form a logical xy-plane. In practice, it is important to keep the compensation distance small in order to reduce the overhead in the switching mechanisms. The same with previous works, d is also limited to 1 in this paper, which keeps the low cost of physical implementation. *xz-plane rerouting scheme* and *yz-plane rerouting scheme* can be obtained in the similar manner.

Under the xy-plane rerouting scheme with compensation distance d of 1, a PE, say e, can directly connects to 12 physically adjacent PEs, as shown in Fig. 2 (c). The 12 PEs distribute in four directions, i.e., x-, $x+$, y- and $y+$ direction. The e's adjacent set of fault-free PEs lying in $y+$ direction of PE e, denoted as $Adj_{y+}(e)$, is defined as

$$\{\ u|u \text{ is fault-free },X(u) = X(e), Y(u) = Y(e) + 1, |Z(u) - Z(e)| \leq 1\ \}.$$

$Adj_{y-}(e)$, $Adj_{x-}(e)$ and $Adj_{x+}(e)$ can be defined similarly. An example of $Adj_{y+}(e)$ is demonstrated in Fig. 2(c). To form a logical plane, PE e needs to connect to one PE from each of its 4 adjacent set. PE e is said to be ***unconnectable*** if any one of $|Adj_{y+}(e)|$, $|Adj_{y-}(e)|$, $|Adj_{x+}(e)|$ or $|Adj_{x-}(e)|$ is 0, where $|A|$ represents the number of elements in set A. This is because, under such a situation, e cannot be used to form a logical plane in this case. In Fig. 2 (c), PE e is unconnectable since $|Adj_{y-}(e)|$=0. In addition, the PE of smallest z-axis index in $Adj_{y+}(e)$ is called the *lowest* fault-free PE in $Adj_{y+}(e)$.

2.2 Previous Works and Problem Description

The problem of constructing maximum logical logical array on selected indexes, investigated in [18], can be formed as follows.

Problem $\mathcal{R}$: *Given an $m \times n \times h$ host array H with faults, x-axis index set* `ROWs` *and y-axis index set* `COLs`*, find a maximum logical array such that each logical plane of the logical array contains exactly one fault-free PE from each $H_{x,y}$ for $x \in$* `ROWs`*, $y \in$* `COLs`*, under $(x \cup y \cup z)$-bypass and xy-plane rerouting scheme.*

Note that if the i-th yz-plane contains too many faulty PEs, then the index x_i will not be included in `ROWs` so that the entier yz-plane will be bypassed using x-direction bypass. Similarly, if the j-th xz-plane contains too many faulty PEs, then the index y_j will not be included in `COLs`.

To construct 3D logical arrays, the algorithm `GPR` in [18] first constructs several logical xy-planes using xy-plane rerouting, then connects these logical planes in z direction. These logical planes are the same size and are constructed on selected x-axis indexes and y-axis indexes. Each logical plane is formed as follows. Assume `ROWs` $=\{x_1, x_2,..., x_s\}$, `COLs`$=\{y_1, y_2,..., y_t\}$ are the selected x-axis indexes and y-axis indexes, respectively. For simplicity, we use px to denote the index prior to x in `ROWs`, use nx to denote the index next to x in `ROWs`, use py to denote the index prior to y in `COLs`, and use ny to denote the index next to y in `COLs`.

In forming the k-th logical plane T_k, `GPR` starts from the position (x_1, y_1), and tries to find a lowest fault-free PE for $e'_{i,j,k}$ $(x_1 \le i \le x_s, y_1 \le j \le y_t)$ from the set S=$Adj_{y+}(e'_{x,py,k}) \cap Adj_{r+}(e'_{p,px,y}) \cap H_{x,y}$, in the manner from x_1 to x_s, from y_1 to y_t. At each position (i, j), if a feasible fault-free PE is found, it switches to the next position; otherwise, it backtracks to its prior column or prior row according to different conditions. This process repeats until 1) a fault-free PE is found to be $e'_{x_s,y_t,k}$ for the k-th logical plane T_k, or 2) the algorithm backtracks to the position (x_1, y_1). The detailed description of algorithm `GPR` is omitted due the limitation on paper length.

Fig. 3 shows nine possible types of link-ways for a feasible logical array. Fig. 3 (a), (b) and (c) are short interconnects [13,18], as they use only one switch to connect 2 PEs. Fig. 3 (d), (e), (f), (g), (h) and (i) are long-interconnects as they use two switches to connect two logically neighboring PEs. Clearly, long-interconnect leads to extra communication delay, capacitance and dynamic power dissipation. Algorithm `GPR` always constructs down-most logical planes in order to find a

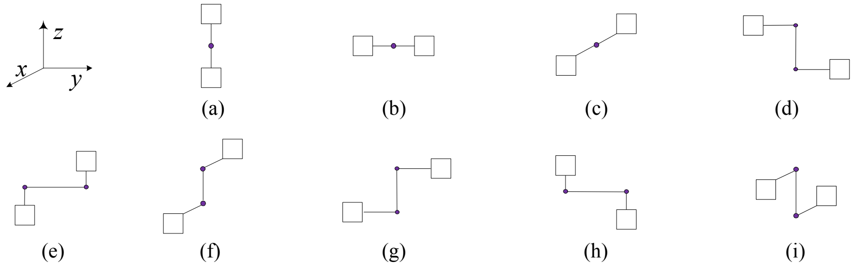


Fig. 3. Short interconnects and long-interconnects

logical array as large as possible. However, down-most planes contain large number of long-interconnects. Thus, it is necessary to reduce long-interconnects for the logical array by `GPR`.

A maximum logical array constructed by algorithm GPR is called *GPR array*. A *communication efficient logical array* (*CELA*) is a maximal sized logical array of the minimum number of the long-interconnects.

The work in [18] also considered the problem of reducing the interconnection length, i.e. constructing *CELA*. The algorithm `CAR` in [18] revises each logical plane of the GPR array, from top to bottom, to reduce the number of long interconnects. Let T_1, T_2, $\cdots$, T_k be the k logical planes of a GPR array. `CAR` revises these planes in the order of T_k, T_{k-1}, ..., T_1, to form better planes. In the process of revising logical plane T_p on an area, say Λ, `CAR` examines each position (i, j) $(x_1 \leq i \leq x_s, y_1 \leq j \leq y_t)$, in the manner from x_1 to x_t, from y_1 to y_s. At each position (i, j), `CAR` selects the best candidate PE for $e'_{i,j,p}$ from set $S_{i,j} = Adj_{y+}(e'_{i,pj,p}) \cap Adj_{x+}(e'_{pi,j,p}) \cap \Lambda$ according to local structure information. For more details see [18].

3 Hardness of Reducing the Interconnection Length

To show that minimizing the inter-length of the MLA is NP-hard, we only need to prove that a special case, i.e., minimizing the inter-length of a single plane (denoted as $\mathcal{MP}$), is NP-hard. The special case $\mathcal{MP}$ is formed as follows,

Problem $\mathcal{MP}$***:*** *Given an area* $\Lambda = \cup \Lambda_{x,y}$ $(x \in \texttt{ROWs}, y \in \texttt{COLs})$ *where* $\Lambda_{x,y}$ *is a subset of* $H_{x,y}$ *such that* $|\Lambda_{x,y}| \geq 1$, *find one PE from each* $\Lambda_{x,y}$ *to form a logical plane such that the inter-length of the logical plane is minimized.*

In a $m \times n$ logical plane, say T_a, there exists $(m-1) \times n + m \times (n-1)$ interconnects in total. The inter-length of logical plane T_a, denoted as $Len(T_a)$, can be calculated by summing up the length of these interconnects. Formally, $Len(T_a) = \sum Len(u, v)$, where u and v are logical neighbors in T_a and $Len(u, v)$ is the length of the interconnect between u and v. On the other hand, if u and v are not logical neighbors, then there is no direct interconnect between the PEs, thus $Len(u, v) = 0$. Therefore, the inter-length of T_a can also be calculated as

$Len(T_a) = \sum_{u,v \in T_a} Len(u,v)$. Problem $\mathcal{MP}$ is form a logical plane T_a such that $Len(T_a) = \sum_{u,v \in T_a} Len(u,v)$ is minimized.

***Theorem* 1**. Problem $\mathcal{MP}$ is NP-hard.

Proof: Here is the polynomial time reduction from the independent set problem, which is a well known NP-hard problem. Given a graph G with vertex set $V = \{v_1, v_2, ..., v_k\}$ and integers m, n ($m \times n < k$), we create an instance of problem $\mathcal{MP}$, consisting of $m \times n$ copies of V as

$$\Lambda_{x,y} = \{v_{x,y,1}, v_{x,y,2}, ..., v_{x,y,k}\}, 1 \leq x \leq m \text{ and } 1 \leq y \leq n.$$

For any two nodes $v_{x_1,y_1,i}$ and $v_{x_2,y_2,j}$, there exists an interconnect whose length is denoted as $Len(v_{x_1,y_1,i}, v_{x_2,y_2,j})$. The length $Len(v_{x_1,y_1,i}, v_{x_2,y_2,j})$ is defined as follows.

If $i \neq j$ and there is no edge between v_i and v_j in G, then $Len(v_{x_1,y_1,i}, v_{x_2,y_2,j})$ = 0; otherwise, $Len(v_{x_1,y_1,i}, v_{x_2,y_2,j})$=1.

We next show that G has an independent set of $m \times n$ vertices *if and only if* there are $m \times n$ PEs forming a logical plane, say T_a, such that $Len(T_a) = \sum_{u,v \in T_a} Len(u,v) = 0$.

1. *On one hand*, suppose that there exists a logical plane T_a such that $Len(T_a) = 0$. Let v_{x_1,y_1,z_1}, v_{x_2,y_2,z_2}, ..., v_{x_i,y_i,z_i}, ... ($1 \leq x_i \leq m, 1 \leq y_i \leq n$) be the PEs in T_a, we can construct a independent set IS as $\{v_{z_1}, v_{z_2}, ..., v_{z_i}, ...\}$ ($1 \leq z_i \leq m \times n$). Clearly, for any two nodes in IS, say v_i and v_j, there is no edge between v_i and v_j in G as $Len(v_{x_i,y_i,i}, v_{x_j,y_j,j})$=0. Thus, IS is a independent set.
2. *On the other hand*, suppose that G has an independent subset IS containing $m \times n$ vertices of G ($m \times n < k$), i.e., $v_{z_1}, v_{z_2}, ..., v_{z_i}, ...$ ($1 \leq i \leq m \times n$). We can construct a logical plane T_a by selecting one PE from each $\Lambda_{x,y}$ ($1 \leq x \leq m, 1 \leq y \leq n$) such that, for any two PEs $v_{x_i,y_i,i}, v_{x_j,y_j,j}$ in T_a, we have $i \neq j$. It is clearly $Len(v_{x_i,y_i,i}, v_{x_j,y_j,j})$=0, because $i \neq j$ and there exists no edge between v_i and v_j in G. Thus, $Len(T_a) = \sum_{u,v \in T_a} Len(u,v)$ = 0.

It is easy to see that the size of the obtained problem is at most k^2 since $(m \times n) < k$, thus the reduction can be down in polynomial time. Therefore, we conclude that Problem $\mathcal{MP}$ is NP-hard.

4 The Proposed Algorithm

Before introducing the proposed algorithm, we first present some notations. Let T_i and T_j be two logical planes constructed on index sets `ROWs` and `COLs`, where `ROWs`$=\{x_1, x_2, ..., x_s\}$ and `COLs`$=\{y_1, y_2, ..., y_t\}$. Let $e'_{x,y,i}$ and $e'_{x,y,j}$ ($x \in$ `ROWs`$, y \in$ `COLs`) denote the PE located at (x, y) of plane T_i and T_j, respectively. The partial order between on T_i and on T_j is defined as follows.

(1) *plane $T_i < T_j$ if PE $e'_{x,y,i}$ of T_i lies to the downside of PE $e'_{x,y,j}$ of T_j, for* $x \in$ `ROWs`$, y \in$ `COLs`.

(2) *plane* $T_i \leq T_j$ *if PE* $e'_{x,y,i}$ *of* T_i *lies to the downside of, or is identical to, the PE* $e'_{x,y,j}$ *of* T_j*, for* $x \in \texttt{ROWs}, y \in \texttt{COLs}$.

If plane $T_d \leq T_i$ for any logical plane T_i that can be constructed from H on `ROWs` and `COLs`, then T_d is called the down-most logical plane.

We use $\Lambda = \mathcal{A}[T_i, T_j)$ $(T_i \leq T_j)$ to indicate the area that consists of the fault-free PEs bounded by T_i and T_j (including T_i but not including T_j). T_i and T_j are called the bottom boundary of $\mathcal{A}[T_i, T_j)$ and the top boundary of $\mathcal{A}[T_i, T_j)$, respectively. Let $\Lambda_{i,j}$ be a subset of Λ such that each PE in $\Lambda_{i,j}$ is of x-axis i and physical y-axis j.

Assume algorithm `GPR` produces a logical array consisting of L logical planes, i.e., T_1, T_2, ..., T_L, named from from bottom to top. Our proposed algorithm `CAAE` revises each logical plane $T_k (1 \leq k \leq L)$, in top-to-bottom manner.

Explicitly, `CAAE` starts from revising T_L on the area $\Lambda = \mathcal{A}[T_L, \infty)$ consisting of fault-free PEs that are located to the up side of plane T_L (including PEs on T_L), resulting in a better logical plane T'_L. Then `CAAE` refines T_{L-1} to a better one, say T'_{L-1}, on the area $\mathcal{A}[T_{L-1}, T'_L)$, followed by refining T_{L-2} to T'_{L-2} on $\mathcal{A}[T_{L-2}, T'_{L-1})$, ..., refining C_1 to C'_1 on $\mathcal{A}[C_1, C'_2)$. It is evident that on every area Λ, one and only one logical plane can be constructed.

Algorithm `CAAE` performs the following three steps on an area Λ to form a better logical plane.

1. `CAAE` first recursively excludes unconnectable PEs out of area Λ. It examines each PE e in area Λ, if any one of $|Adj_{y+}(e)|$, $|Adj_{y-}(e)|$, $|Adj_{x+}(e)|$ or $|Adj_{x-}(e)|$ is 0, then PE e is identified to be unconnectable and will be excluded out of set Λ. Once a PE, say $e_{i,j,k}$, is identified as unconnectable PE, all its physical neighbors (12 at most) will be re-examined.
2. `CAAE` calculates the global statistical information of Λ. $XY\_d(i,j)$ is calculated by $|\Lambda_{i,j}|$ for $x_1 \leq i \leq x_s, y_1 \leq j \leq y_t$. Among all PEs in Λ, the number of PEs, whose physical z-axis index is t, is calculated as $Z\_d(t)(1 \leq t \leq h)$. The physical plane of maximum $T\_d(t)$ is selected as the central plane, denoted as cP, to assist revising T_k in next step.
3. In this step, algorithm `CAAE` examines each position $(i,j)(i \in \texttt{ROWs}, j \in \texttt{COLs})$ to select the best PE in $\Lambda_{i,j} = \Lambda_{i,j}$ for inclusion into the logical plane T_k. `CAAE` examines each position in the following order. It first examines positions (i,j) of $XY\_d(i,j) = 1$, then examines positions of $XY\_d(i,j) = 2$,..., and the process repeats until all positions are examined. At position (i,j), the algorithm selects the PE located in physical plane cP from $\Lambda_{i,j}$ for inclusion into T_k, i.e., PE $e_{i,j,cP}$ is selected to be $e'_{i,j,k}$. However, if no PE in $\Lambda_{i,j}$ is located in plane cP, then `CAAE` selects the one mostly close to plane cP from $\Lambda_{i,j}$. If there are two PEs in $\Lambda_{i,j}$ that are of the same distance to plane cP, then `CAAE` selects the one of larger z-axis index. After selecting one from $\Lambda_{i,j}$, all unselected ones are excluded out of Λ.

We now analyze the time complexity of the proposed algorithm for revising a $m \times n \times L$ logical array constructed from a $m \times n \times h$ host array. Let θ_k $(1 \leq k \leq L)$ denote the number of PEs in the area Λ for revising logical plane T_k, thus θ_k is bounded by $(h - L) \times m \times n$, where L is the number of logical

Algorithm 1. CAAE

```
Input: host array $H$ with size of $m \times n \times h$; logical array $T$, number of logical
       planes $L$;
Output: optimized logical array $T$;
begin
$k \leftarrow L$;    /* $L$ is number of logical planes in $T$ */
while $k > 0$ do
    $\Lambda \leftarrow \mathcal{A}[T_k, T_{k+1}]$;    /* set optimization area */
    step 1): /* exclude unconnectable PEs out of $\Lambda$ */
    for each PE $e$ in area $\Lambda$ do
        if $e$ is unconnectable then
            exclude $e$ out of $\Lambda$; Check_Neighbors_Connectivity($e, \Lambda$);
    step 2): /* compute the distribution of PEs in $\Lambda$ */
    $XY\_d(i,j) \leftarrow \mid \Lambda_{i,j} \mid$; $d(t) \leftarrow \mid \{e \mid e \in \Lambda \wedge Z(e) = t\} \mid$;
    $cP \leftarrow \arg\max_{1 \le t \le h} d(t)$.
    step 3): /* select one PE from each $\Lambda_{i,j}$ to form a better $T_k$ */
    $N_c \leftarrow 0$;   /* number of candidate PEs */
    $N_p \leftarrow |\texttt{ROWs}| \times |\texttt{COLs}|$;    /* number of unchecked positions */
    while $N_p > 0$ do
        $N_c \leftarrow N_c + 1$;
        for each $(i,j)$  $(i \in \texttt{ROWs}, j \in \texttt{COLs})$ do
            if $(i,j)$ is unexamined and $XY\_d(i,j) = N_c$ then
                $e'_{i,j,k} \leftarrow$ the PE $e$ $(e \in \Lambda_{i,j})$ that closest to plane $cP$;
                mark position $(i,j)$ as examined;   $N_p \leftarrow N_p - 1$;
                exclude unselected PEs of $\Lambda_{i,j}$ out of $\Lambda$;
    $k \leftarrow k - 1$;
return $T$;
end
```

```
Procedure Check_Neighbors_Connectivity($e_0, \Lambda$);
/*Recursively check the connectability of $e_0$'s neighbors in $\Lambda$ */
begin
put $e_0$ in queue $Q$;
while ($Q$ is not empty) do
    $p \leftarrow$ the first element in $Q$;
    for each $e$ of $p'$s physical neighbors in $\Lambda$ do
        if ($e$ is unconnectable) then
            exclude $e$ out of $\Lambda$;   Put $e$ into queue $Q$;
```

planes. In revising T_k: step (1) runs in $O(\theta_k)$; step (2) runs in $O(\theta_k)$; step (3) runs in $O((h-L)\times m\times n)$ because the while loop runs $h-L$ times at most and each loop runs in $O(m\cdot n)$. Thus, the process of revising one logical plane takes $O(\theta_k)$+$O(\theta_k)$+$O((h-L)\times m\times n)$=$O((h-L)\times m\times n)$ time. Therefore, we conclude that the time complexity of CAAE is $O(L\times(h-L)\times m\times n)$, as there are L logical planes in total.

5 Experimental Results and Analysis

In this section, we experimentally investigate the efficiency of our proposed algorithm CAAE in comparison with previous algorithms GPR and CAR. The three algorithms were implemented in C++ language and run 2.1 GHz CPU with 4 GB RAM. In order to make a fair comparison, we keep the same assumptions as in [6, 8-18]. All algorithms were tested and compared on datasets generated in the same way as in previous work. The fault distribution in the entire mesh is in a uniform manner, which corresponds to the fault distribution of a random fault model.

Two evaluation metrics, *harvest* (*harv*) and the number of long-interconnects (*nlis*), are calculated. The *harv* indicates how effectively the non-faulty elements are utilized in constructing a logical array from a host array with fault elements [12,13,18].

$$harv = \frac{\text{Size of logical array } T}{\text{number of nonfaulty PEs in host array } H}\times 100\%$$

The *nlis* reflects the interconnection redundancy of the produced logical array. Let GPR_*nlis* denotes the *nlis* of a GPR array and CAR_*nlis* denotes the *nlis* of an logical array produced by algorithm CAR. Let New_*nlis* denotes the *nlis* of an logical array produced by our proposed algorithm CAAE. Then the improvement of CAAE over GPR in terms of *nlis*, denoted as *imp.G*, is calculated by

$$imp.G = (1-\frac{\texttt{New\_}nlis}{\texttt{GPR\_}nlis})\times 100\%$$

And the improvement of CAAE over CAR in terms of *nlis*, denoted as *imp.C*, is calculated by

$$imp.C = (1-\frac{\texttt{New\_}nlis}{\texttt{CAR\_}nlis})\times 100\%$$

Table 1 shows performance comparison of algorithms GPR, CAR and the proposed algorithm CAAE. The fault density of the host array ranges from 0.1% to 5%. From table 1, it can be seen that algorithm CAAE clearly outperforms algorithms GPR and CAR in terms of *nlis* without loss of harvest. This is because algorithm CAAE revises each logical planes using fault-free PEs in dynamically calculated areas. Each area is bounded by two logical planes, which guaranteeing that revising a logical plane to a better one does not occupy PEs that are critical for generating other logical planes. In addition, the global statistical information

Table 1. Performance evaluation on host arrays with size ranges from 32 × 32 × 32 to 128 × 128 × 128, averaged over 20 instances for each case

Host Array		Logical Array		Interconnection				
size	ρ(%)	size	*harv* (%)	GPR_*nlis*	CAR_*nlis*	New_*nlis*	*imp.G* (%)	*imp.C* (%)
32×32×32	0.1	32×32×31	96.3	1857	2148	931	49.8	56.6
	1	32×32×29	90.6	15531	11776	6430	58.6	45.4
	3	32×32×27	86.0	26004	14887	11899	54.2	20.1
	5	32×32×24	78.6	26577	16410	12892	51.5	21.4
64×64×64	0.1	64×64×62	97.0	31178	32862	15578	50.0	52.6
	1	64×64×59	92.8	202748	130012	82724	59.2	36.4
	3	64×64×53	85.9	246215	156513	122589	50.2	21.7
	5	64×64×49	80.3	238712	152389	126982	46.8	16.7
96×96×96	0.1	96×96×93	97.2	159114	147263	84936	46.6	42.3
	1	96×96×89	93.3	805482	535772	342265	57.5	36.1
	3	96×96×81	87.0	886517	574549	465330	47.5	19.0
	5	96×96×74	81.1	840519	560615	477129	43.2	14.9
128×128×128	0.1	128×128×125	97.6	501963	454638	267636	46.7	41.1
	1	128×128×119	93.8	2068040	1359890	931239	55.0	31.5
	3	128×128×109	87.8	2177390	1463362	1201480	44.8	17.9
	5	128×128×101	82.6	2064600	1404370	1232390	40.3	12.2

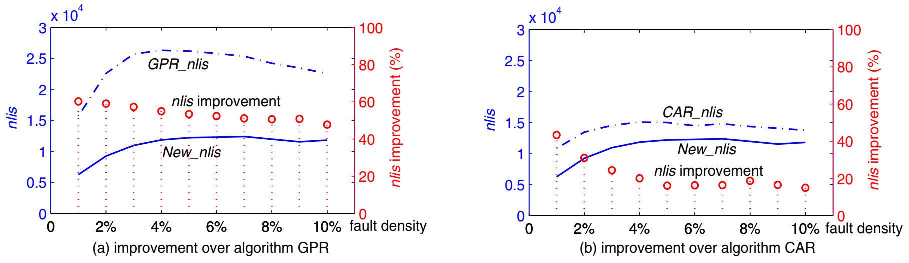


Fig. 4. Performance comparison of algorithms GPR, CAR and the the proposed algorithm CAAE on 32 × 32 × 32 host array with different fault densities, averaged over 20 instances for each case

is utilized in refining a logical planes which ensures better *nlis* performance. In general, for all cases considered in this paper, the average improvement of algorithm CAAE over GPR is is about 49.7% and the average improvement of algorithm CAAE over CAR is is about 29.8%.

Next, we analyze the impact of fault density on the proposed algorithm. Fig. 4(a) shows the performance comparison between algorithm GPR and CAAE in terms of *nlis*, on 32×32×32 host arrays with fault density ranges from 1% to 10%. Under a certain point, *nlis* increases with the increased fault density for both algorithms; then, a decrease in the values of *nlis* is observed on arrays with larger fault density. The increase in *nlis* for both algorithms, on host arrays with smaller fault densities, is due to the fact that the increasing fault density reduces the chance of constructing communication-efficient logical array (*CELA*). On the other hand, the decrease in *nlis*, on host arrays with larger fault densities, is due to a considerable reduction in logical array size. Also shown in Fig. 4(a), the improvement of the algorithm CAAE over GPR in terms of *nlis* increases with

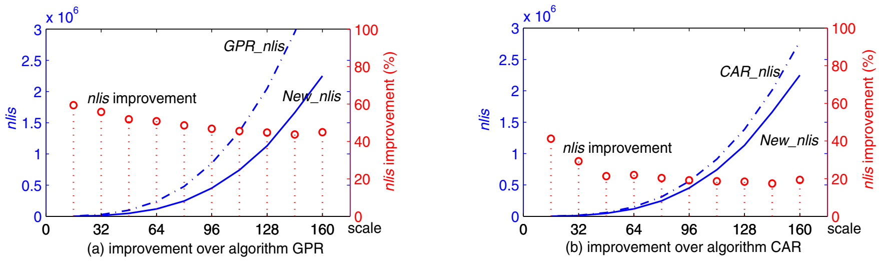


Fig. 5. Performance comparison of algorithms GPR, CAR and the the proposed algorithm CAAE on host arrays of different scales with 5% faulty PEs, averaged over 20 instances for each case

increasing fault density on less defective host arrays; then it decreases when the fault density becomes higher. This is because the algorithm GPR always selects the lowest fault-free PEs to form logical planes, which resulting in producing long-interconnects around the faulty PEs. Thus, up to a certain point, the increase in fault density leads to a larger number of long-interconnects in a GPR array. This provides more chances for optimization by algorithm CAAE, resulting in better improvement in *nlis*. On the other hand, beyond a certain point of fault density, the number of unused PEs to be used for optimizing logical array decreases with the increasing fault density, thus leads to lower improvement on *nlis*. Generally, the average improvement is 53.7% on $32 \times 32 \times 32$ host arrays. Similar results are obtained in the comparison between algorithm CAAE and CAR as shown in Fig. 4(b).

Fig.5(a) shows the comparison between algorithms GPR and CAAE on host arrays with size ranging from $16 \times 16 \times 16$ to $160 \times 160 \times 160$. It is observed that the *nlis* increases with increasing array size for both GPR and CAAE, because the size of logical array grows with the increasing host array size. The improvement of *nlis* experiences an increment with the increase of host array size on small or medium-sized host arrays, and then it decreases on large scale host arrays. The increase in the improvement on small host arrays is caused by *long-interconnect transfer*; that is, if a logical plane contains some long-interconnects, then its neighboring logical plane will possibly contains long-interconnects at the same position. These transferred long-interconnects can be optimized by algorithm CAAE. The increasing host array size results in the increase in the number of logical planes, thus produces more transferred long-interconnects which provides more chances for optimization by CAAE. Therefore, the improvement of CAAE over GPR in *nlis* increases with increasing host array size. On the other hand, beyond a certain point, the impact of the long-interconnect transfer remains stable while the value of *nlis* grows rapidly with increasing host array size, thus resulting in the decrease in the improvement. Similar results are obtained in the comparison between algorithm CAAE and CAR as shown in Fig. 5(b).

6 Conclusions

In this paper, we have investigated the problem of constructing communication efficient logical array(*CELA*) for 3D fault-tolerant processor arrays. We proved that reducing the interconnection redundancy is NP-hard. We have presented novel heuristic to reduce the long-interconnects by revising each logical plane of the GPR array. Unlike the existing algorithms, we optimize the interconnection length by revising the logical array based on its structure information. Experimental results show that the proposed algorithm is superior in both runtime complexity and interconnection length.

Acknowledgment. This work was supported by the National Natural Science Foundation of China under Grant No. 61173032 and No. 61303021, and the Doctoral Fund of Ministry of Education of China under Grant No. 20100032110041 and No. 20131201110002.

References

1. Koren, I., Koren, Z.: Defect tolerance in VLSI circuits: Techniques and yield analysis. Proceedings of the IEEE 86(9), 1819–1838 (1998)
2. Koren, I.: On the design and analysis of fault tolerant NoC architecture using spare routers. In: Proc. IEEE Int. Symp. on Quality Electronic Design, pp. 115–120 (2000)
3. Tan, P.J., Le, T., Mantri, P., Westfall, J.: Testing of UltraSPARC T1 Microprocessor and its Challenges n. In: Proc. IEEE Int. Test Conf, pp. 1–10 (2006)
4. Makar, S., Altinis, T., Patkar, N., Wu, J.: Testing of Vega2, a chip multi-processor with spare processors. In: Proc. IEEE Int. Test Conf, pp. 1–10 (2007)
5. Schuchman, E., Vijaykumar, T.: Rescue: A microarchitecture for testability and defect tolerance. In: Proc. Int. Symp. on Computer Architecture, pp. 160–171 (2005)
6. Ajwani, D., Ali, S., Morrison, J.: Graph Partitioning for Reconfigurable Topology. In: IEEE Int. Parallel & Distributed Processing Symp., Shanghai, China, pp. 836–847 (2012)
7. Modarressi, M., Sarbazi-Azad, H., Tavakkol, A.: An efficient dynamically reconfigurable on-chip network architecture. In: Proc. Design Automation Conference, pp. 310–313 (2010)
8. Modarressi, M., Tavakkol, A., Sarbazi-Azad, H.: Application-Aware Topology Reconfiguration for On-Chip Networks. IEEE Trans. on VLSI Systems 19(11), 2010–2022 (2011)
9. Modarressi, M., Sarbazi-Azad, H.: Power-aware mapping for reconfigurable NoC architectures. In: Proc. of the 25th International Conference on Computer Design, pp. 417–422 (2007)
10. Takanami, I., Horita, T.: A Built-in Circuit for Self-Repairing Mesh-Connected Processor Arrays by Direct Spare Replacement. In: IEEE Pacific Rim Int. Symp. on Dependable Computing (PRDC), Niigata, pp. 96–104 (2012)
11. Lin, S.Y., Shen, W.C., Hsu, C.C., Fault-tolerant, A.Y.W.: Router with built-in self-test/self-diagnosis and fault-isolation circuits for 2D-mesh based chip multiprocessor systems. Jour. of Electrical Engineering 16(3), 213–222 (2009)

12. Low, C.P.: An efficient reconfiguration algorithm for degradable VLSI/WSI arrays. IEEE Transactions on Computers 49(6), 553–559 (2000)
13. Wu, J., Thambipillai, S., Jiang, G., Wang, K.: Constructing Sub-Arrays with Short Interconnects from Degradable VLSI Arrays. IEEE Transactions on parallel and Distributed System 25(4), 929–938 (2014)
14. Wu Jigang, T., Srikanthan, X., Han, X.: Preprocessing and Partial Rerouting Techniques for Accelerating Reconfiguration of Degradable VLSI Arrays. IEEE Transactions on Very Large Scale Intergration (VLSI) Systems 18(2), 315–319 (2010)
15. Fukushi, M., Fukushima, Y., Horiguchi, S.: A genetic approach for the reconfiguration of degradable processor arrays. In: Proc. of 20th IEEE International Symposium on Defect Fault Tolerance VLSI System, pp. 63–71 (2005)
16. Takanami, I., Horita, T.: A Built-in Self-Reconfigurable Scheme for 3D Mesh Arrays. In: International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 1997), pp. 458–464 (1997)
17. Horiguchi, S., Numata, I.: Self-Reconfiguration Scheme of 3D-Mesh Arrays. In: Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 276–281 (1998)
18. Jiang, G., Wu, J., Sun, J.: Efficient Reconfiguration Algorithms for Communication-Aware Three-dimensional Processor Arrays. Parallel Computing 39(9), 490–504 (2013)
19. Fukushi, M., Horiguchi, S.: A Self-reconfigurable Hardware Architecture for Mesh Arrays Using Single/double Vertical Track Switches. IEEE Trans. on Instrumentation and Measurement 53(2), 357–367 (2004)

Feature Evaluation for Early Stage Internet Traffic Identification

Lizhi Peng[1,2], Hongli Zhang[1], Bo Yang[2], and Yuehui Chen[2]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150002, P.R. China
[2] Provincial Key Laboratory for Network Based Intelligent Computing, University of Jinan, Jinan, 250022, P.R. China
{plz,yangbo,yhchen}@ujn.edu.cn, zhanghongli@hit.edu.cn

Abstract. Identifying a network traffic at its early stage accurately is very important for the application of traffic identification. And this has caught a lot of interests in recent years. Packet sizes and statistical features are effective features that widely used in early stage traffic identification. However, an important issue is still unconcerned, that is whether there exists essential differences between using the packet sizes and derived features such as statistics in early stage traffic identification. In this paper, we set out to evaluate the effectiveness of different kinds of early stage traffic features. We firstly extract the packet sizes and their derived features of the first 10 packets on 3 traffic data sets. Then the mutual information between each feature and the corresponding traffic type label is computed to show the effectiveness of the feature. And then we execute a set of crossover identification experiments with different feature sets using 7 well-known classifiers. Our experimental results show that most classifiers get almost the same performances using packet sizes and derived features for early stage traffic identification. And the combined feature set selected by mutual information can obtain high identification performances.

Keywords: Feature selection, Early stage traffic classification, Machine learning.

1 Introduction

In recent years, early stage traffic identification has caught enough interests at the research community. Most traditional machine learning based traffic identification techniques extract features on a whole traffic instance [7], [13], [15]. The most widely used feature extracting method is presented by A. W. Moore et al. in 2005 [14]. They extract 248 statistical features based on a whole traffic, such as maximum, minimum and average values of packet size, RTT. And classifiers using these statistical features can get very high performances in traffic identification. However, in real circumstances, it makes no sense to recognize Internet traffics when they have ended. Thus, we must identify Internet traffics accurately in their early stage so that we can apply subsequent management and security policies. Therefore, some researchers have turned to find effective models which are able to identify Internet traffics at their early stage. And this makes early stage identification to become a hot topic in traffic identification researches [3]. B. Qu et al. have studied the problem of accuracy of early stage traffic identification, and found that it is possible to identify traffic accurately at its early stage [19].

It is relatively hard to recognize a traffic by only using several early stage packets. According to A. Dainotti, limiting the number of packets used to extract features offers several benefits including lower feature extraction complexity [3]. However, are the simple features extracted based on so few packets effective enough for identification? Thus, the key problem of early stage traffic identification is to find out effective features in the early stage of a traffic. L. Bernaille et al. presented a famous early stage traffic identification technique in 2006 [1]. They use the sizes of the first few data packets of each TCP flow as the features, and by applying K-means clustering technique, they got high identification rates for 10 types of application traffics. A. Este et al. have proved in 2009 [6] that the early stage packets of an Internet flow carry enough information for traffic classification. They analyzed round trip time (RTT), packet size, inter-arrival time (IAT) and packet direction of the early stage packets and found that the packet size is the most effective feature for early stage classifications. N. Huang et al. have studied the early stage application characteristics and used them for classification effectively in 2008 [8]. Recently, they extracted a set of early stage traffic features by analyzing the negotiation behaviors of different applications. They use the packet size (PS) and the inter packet time (IPT) of the first 10 packets for some classifiers, while for other classifiers, they use the average and the standard deviations of PS and IPT of the early packets. They applied these features for machine learning based classifiers with high performances [9]. B. Hullár et al. proposed an automatic machine learning based method consuming limited computational and memory resources for P2P traffic identification at early stage [11]. A. Dainotti et al. [4] construct high effective hybrid classifiers and apply a hybrid feature extraction method for early stage traffic classification. T. T. T. Nguyen et al. use statistical features derived from sub-flows for timely identification of VoIP traffics [16], they extend the concept of early stage to “timely”, since a sub-flows refers to a small number of most recent packets taken at any point in a flows lifetime.

For the studies mentioned above, packet level features or derived features such as statistics, were applied to identify Internet traffics. Features are designed empirically by researchers. However, the effectiveness of different kinds of features of the early stage is unknown. The packet level features are able to show the detailed characteristics of an Internet traffic, while they can not catch its global characteristics. On the contrary, a statistical feature such as the average packet size is able to show the global distribution characteristics of a traffic. However, the number of packets in the early stage of a traffic is considerably small, usually, it ranges from 4 to 10. Thus, is an early stage feature able to include enough information for identification, and are derived early stage features more effective than packet level features? These questions should be answered.

Contributions: In this paper, we set out to study the effectiveness of the early stage features of Internet traffics. We try to answer the above mentioned questions using mutual information analysis and experimental methods. 3 traffic data sets and 7 machine learning classifiers are applied for our experiments. We use the application layer packet sizes as the original packet level features. The first and second differences and six statistics are applied as the derived features. Firstly, the mutual information of each feature and the traffic type label is computed to evaluate its effectiveness preliminary. Then we build 5 feature sets covering the pure original feature set, the pure derived feature set and the hybrid feature set, and then all selected classifiers are applied using these feature sets to validate the effectiveness of selected features.

The rest of the paper is organized as follows: Section 2 introduces the characteristics of the selected data sets applied in the study. All features to be evaluated are presented in Section 3. And Section 4 illustrates the methods applied in our study. The details of experimental results and analysis are given in Section 5. Finally, we draw some conclusions in Section 6.

2 Data Sets

We select two sets of open network traffic traces, and a set of traces collected in our campus network for our study. The characteristics of the selected traces are depicted in Table 1.

Table 1. Characteristics of the selected network traffic traces

Auckland II traces			UNIBS traces			UJN traces		
Type	#inst	Bytes	Type	#inst	Bytes	Type	#inst	Bytes
ftp	251	136241	bittorrent	3571	6393487	Web Browser	11890	58025350
ftp-data	463	5260804	edonkey	379	241587	Chat	11478	60212804
http	23721	139421961	http	25729	107342346	Cloud Disk	1563	109552924
imap	193	86455	imap	327	860226	Live Update	2169	28759962
pop3	498	98699	pop3	2473	4292419	Stream Media	810	785556
smtp	2602	1230528	skype	801	805453	Mail	803	2092862
ssh	237	149502	smtp	120	43566	P2P	326	2521089
telnet	37	21171	ssh	23	39456	Other	1408	3635558

2.1 Auckland II Traffic Traces

Auckland II is a collection of long GPS-synchronized traces taken using a pair of DAG 2 cards at the University of Auckland which is available at [22]. There are 85 trace files which were captured from November 1999 to July 2000. Most traces were targeted at 24 hour runs, but hardware failures have resulted in most traces being significantly shorter. We selected two trace files captured at Feb 14 2000 for our study. The traces include only the header bytes, with a maximum amount of 64 bytes for each frame, while the application payload is fully removed. And all IP addresses anonymised using Crypto-Pan AES encryption. The header traces were captured with a GPS synchronized mechanism using a DAG3.2E card connected to a 100Mbps Ethernet hub interconnecting the University's firewall to their border router.

Since the application payloads were not recorded in Auckland II, DPI tools are invalid to obtain ground truths. The only way to pick out the original application type is using port numbers. In this study, we only accounted TCP case since TCP is the predominant transport layer protocol. Each flow is thus assigned to the class identified by the server port. We selected 8 main types from Auckland II traces and filtered mouse flows with no more than 10 non-zero packets as illustrated in Section 2.

2.2 UNIBS Traffic Traces

UNIBS is another opening traffic traces developed by Prof. F. Gringoli and his research team, available at [21]. They developed a useful system namely GT [12] to application ground truths of captured Internet traffics. The traces were collected on the edge router of the campus network of the University of Brescia on three consecutive working days (Sept 30, Oct 1 and Oct 2 2009). They are composed of traffic generated by a set of twenty workstations running the GT client daemon. Traffics were collected by running Tcpdump [20] on the Faculty's router, which is a dual Xeon Linux box that connects the network to the Internet through a dedicated 100Mb/s uplink. 99% flows in UNIBS are TCP flows. Therefore, we again use TCP flows in this data set for our study. By using GT, UNIBS traces recorded the application information of each captured flow. We can get the application ground truths by both TCP port numbers and GT records. We also chose 8 main types in UNIBS for our study which are shown in Table II. Different from Auckland II traces, there are two popular P2P applications in this data set, bittorrent and edonkey, recorded by GT. Skype is also selected as an import Internet application. Flows with no more than 10 non-zero payload packets are also filtered.

2.3 UJN Traffic Traces

The third data set is collected in a laboratory network of University of Jinan using Traffic Labeler (TL) [18]. TL system captures all user socket calls and their corresponding application process information in the user mode on a Windows host, and sends the information to an intermediate NDIS driver in the kernel mode. The intermediate driver writes application type information on the TOS field of the IP packets which match the 5-tuple. By this mean, each IP packet sent from the Windows host carries their application information. Therefore, traffic samples collected on the network have been labeled with the accurate application information and can be used for training effective traffic classification models. We deployed 10 TL instances on Windows user hosts in the laboratory network of Provincial Key Laboratory for Network Based Intelligent Computing. A mirror port of the uplink port of the switch was set, and a data collector was deployed at the mirror port. The deployed TL instances ran at work hours every day. The data collecting process lasted 2 days in May 2013. Again, flows with no more than 10 non-zero payload packets are also filtered.

3 Features

In this study, we use the sizes of the early stage packets as the original packet level features, and derive three types of deuterogenic features based on the packet sizes: the first and the second order differences, and the statistical features. As far as we know, there is no study reported to use differences as the early stage traffic features. So it is very necessary to evaluate their effectiveness.

3.1 Packet Sizes

The packet size has proved to be the most effective packet level feature in the early stage of traffics [6]. We use the packet sizes of the first 10 packets as the original early stage traffic features in this study. All other features are derived from the packet sizes. And we use the abbreviation of *ps* for packet size in this paper.

Choosing an effective early stage packet number is also an interesting problem. As we know, using too many packets will increase the computational complexity of the feature extraction procedures and decrease the efficiency of the identification models. While using too few packets will reduce the identification accuracies since they can not contain enough characteristics of the traffics. Most researchers use empirical values in their studies. In [1], the authors say 5 packets are enough to distinguish the application behaviors. A. Este et al. use the first 6 packets of a flow for their study [6], they also did not say why is 6 packets. In [9], the authors extract early traffic features from 20 packets, and the number is also an empirical value. In this study, we use the first 10 packets empirically. The number is neither too large, nor too small.

3.2 The First and the Second Order Differences

The first order difference of a discrete function is the difference between two adjacent function values. It shows the trends of the function. If we consider the early stage packet sizes of a traffic as a discrete function, then we can get its first order difference as follows:

$$diff1_i = ps_{i+1} - ps_i, i = 1, 2, ..., 9. \tag{1}$$

Where ps_i is the size of the ith packet, and ps_{i+1} is the size of the $i+1$th packet. We select 10 early stage packets and extract their packet sizes. Thus, we get 9 first order difference values. Similarly, the second order difference is the difference between two adjacent first order difference values. And it can be defined as follows:

$$diff2_i = diff1_{i+1} - diff1_i, i = 1, 2, ..., 8. \tag{2}$$

3.3 Statistical Features

Statistical features is widely used for many problems as they can describe the overall characteristics of an object. We select six simple statistical features which are easy to compute.

- **Average:** The average is also known as the arithmetical mean, which is an extensively used statistic. This feature is calculated as follows:

$$avg = \sum_{i=1}^{n} ps_i \tag{3}$$

- **Standard deviation:** The standard deviation shows how much variation or dispersion from the average exists. And the feature is defined as:

$$stdev = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(ps_i - avg)^2} \tag{4}$$

  where n is the number of packets, i. e. 10 in this study.
- **Maximum and minimum:** The maximum and minimum packet size are also applied in the study, and we use the abbreviations of *max* and *min* respectively.
- **Geometric mean:** The geometric mean is another mean which is defined as:

$$gm = \sqrt[n]{ps_1 ps_2 ... ps_n} \tag{5}$$

- **Variance:** The variance measures how far the packet sizes is spread out, which is defined as:

$$var = \frac{1}{n-1}\sum_{i=1}^{n}(ps_i - avg)^2 \tag{6}$$

4 Methodology

4.1 Mutual Information

Mutual information is a useful measure in information theory which is widely used for many research areas. The mutual information of two random variables X, Y is a measure of the variables' mutual dependence. In information theory, mutual information is defined as

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(X,Y) - H(X|Y) - H(Y|X) \end{aligned} \tag{7}$$

where $H(X)$ and $H(Y)$ are the marginal entropies of X and Y respectively, $H(X|Y)$ and $H(Y|X)$ are the conditional entropies, and $H(X,Y)$ is the joint entropy of X and Y. From the point of view of set theory, the relationships

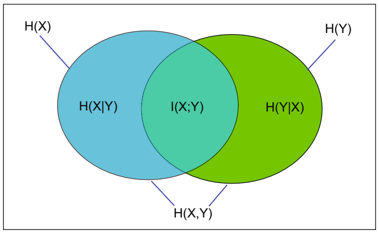


Fig. 1. The relationships among the entropies and the mutual information

among $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$, $H(X,Y)$ and $I(X;Y)$ can be shown as Fig. 1 depicts. According to Shannon's definition of entropy, we have

$$H(X) = -\sum_{x \in X} p(x) log(p(x)) \tag{8}$$

$$H(Y) = -\sum_{y \in Y} p(y) log(p(y)) \tag{9}$$

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) log(p(x,y)) \tag{10}$$

where $p(.)$ is the probability distribution function of a random variable. We use the three equations in equation (7) and can obtain the computational formula of mutual information

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log(\frac{p(x,y)}{p(x)p(y)}) \tag{11}$$

There are many open source software for mutual information computation. And in our study, we apply H. Peng's mutual information Matlab toolbox [17].

4.2 Classifiers

We execute our identification experiments using 7 well-known machine learning classifiers. We use Weka data mining software [23] as our experiment tool. And the selected classifiers fall into five categories according to Weka:

– **Bayes:**Bayes classifiers are based on Bayes theorem, which is widely applied in many engineering areas. In this study, we choose Bayesian network (BayesNet) as the Bayes classifier.
– **Lazy learning:** Strictly speaking, there is no general training procedure for a lazy learning classifier. It just loads the training data in the training phase, and executes real classification decisions in the testing phase. We choose k-nearest neighbor (KNN) classifier for this category.

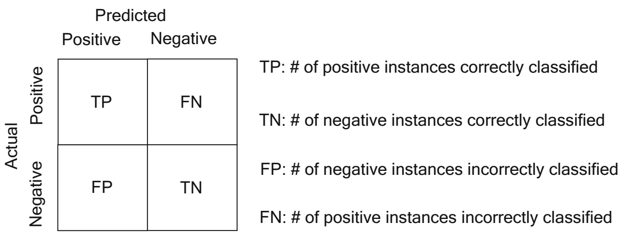

Predicted
Positive
Negative
Actual
Positive
Negative
TP
FN
FP
TN
TP: # of positive instances correctly classified
TN: # of negative instances correctly classified
FP: # of negative instances incorrectly classified
FN: # of positive instances incorrectly classified

Fig. 2. Confusion Matrix

- **Meta:**Strictly speaking, meta classifier is a kind of classification framework based on a specific classifier. This technique firstly trains a group of "weak learn", and then generate a "strong learn" based on the weak learns. We choose Bagging for our study.
- **Rule:** As the name suggests, a rule based classifier extracts rules using a specific policy, e. g. probability and decision trees, and uses the rules to classify testing data. PART is selected for this category in this study.
- **Trees:** This refers to decision trees. A decision tree divides the target feature space hierarchically. Each division produces a node on the decision trees. A classification procedure is a procedure that goes from the root node to a specific leaf node on the tree. In this study, C4.5 decision trees (J48) and random forest (RandomForest) are selected for this category.
- **Functions:** Weka refers all classifiers based on specific functions to this category. We choose logistic regression (Logistic) for this category.

4.3 Performance Measures

The confusion matrix is the basis in measuring a classification task, wherein rows denote the actual class of the instances and the columns denote the predicted class. Fig. 2 shows a typical confusion matrix of a binary classification. We conduct many types of measures based on the confusion matrix to evaluate classifier performance. In this study, the following measures are used:

- **Accuracy:** Classification accuracy (Acc) is defined as the total proportion of all correctly classified instances:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{12}$$

- **Area Under Curve:** The receiver operating characteristic (ROC) curve [2] is a 2D graphical illustration of the trade-off between the TP rate (TPR) and FP rate (FPR). The TPR is also called sensitivity ($Sens$), and the FPR is related to another general measure namely specificity ($Spec$), and they are defined as follows:

$$Sens = \frac{TP}{TP + FN} = TPR, Spec = \frac{TN}{TN + FP} = 1 - FPR \tag{13}$$

The ROC curve illustrates the behavior of a classifier without considering the class distribution or misclassification cost. The area under the ROC curve (AUC) [10] is computed by the confusion matrix values in relation to the TPR and FPR:

$$AUC = \frac{1 + TPR - FRP}{2} = \frac{Sens + Spec}{2} \tag{14}$$

5 Experimental Results and Analysis

5.1 Mutual Information Analysis

We show the the mutual information between each feature and the corresponding traffic type label for each data set in Fig. 3. In the figure, psi is the packet size between the ith packet, $diff1 - i$ is the first order difference of the $i + 1$th and the ith packet sizes, $diff2 - i$ is the ith second order difference.

The packet sizes of the first 2 packets, $diff1 - 1$, and the minimum packet size get the low level mutual information. The results mean that the minimum packet size, the packet sizes of the first 2 packets and their difference contain the fewest identification information. For all of the three data sets, the variance achieves the highest mutual information value, which implies that the variance contribute the most identification information. For Auckland II data set, the mutual information that the packet size feature set, the first order difference feature set and the second order difference feature set contribute are on the similar levels. However, the statistical features make evidently higher mutual information, especially for *avg*, *stdev* and *var*. UNIBS data set shows a pattern which is quite different from that of Auckland II. The packet sizes of the first 10 packets output unstable mutual information. While the mutual information of

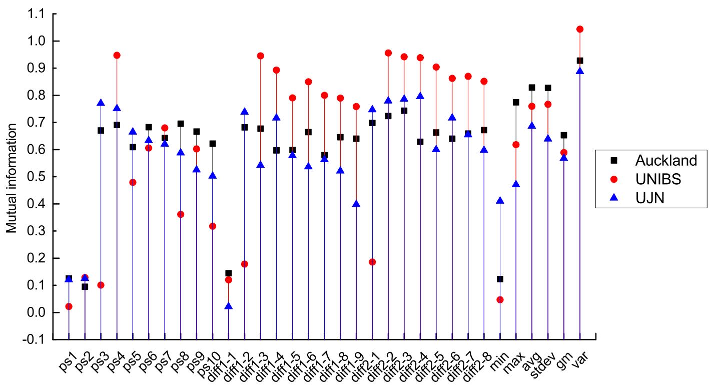


Fig. 3. Mutual information between selected features and traffic type

Table 2. Features selected according to mutual information

Auckland II		UNIBS		UJN	
Ranking	Feature	Ranking	Feature	Ranking	Feature
1	pk4	1	pk4	1	pk3
2	pk6	2	diff1-3	2	pk4
3	pk8	3	diff1-4	3	pk5
4	diff1-2	4	diff1-6	4	diff1-2
5	diff1-3	5	diff2-2	5	diff1-4
6	diff2-1	6	diff2-3	6	diff2-1
7	diff2-2	7	diff2-4	7	diff2-2
8	diff2-3	8	diff2-5	8	diff2-3
9	max	9	diff2-6	9	diff2-4
10	avg	10	diff2-7	10	diff2-6
11	stdev	11	diff2-8	11	avg
12	var	12	var	12	var

the first and second differences are stable, and their overall levels are also higher than that of the packet sizes. Therefore, the first and second differences are effective for UNIBS data set from the point of view of the mutual information. UJN data set also shows a unique pattern. Its mutual information shows a steady decrease from the third packet to the tenth packet. The mutual information that the first and second differences output are not very stable, and the overall level of the second order differences is a little higher than that of the packet sizes.

From a global view, there is no significant differences among the four feature sets with regard to the mutual information. So, the effectiveness of the packet sizes, the first and second differences, and the statistical features are similar according to the mutual information analysis. For each data set, we select 12 features to compose a combined feature set by the ranking of the mutual information of all alternative features. And the selected features are shown in Table 2.

5.2 Identification Results

In this subsection, we carry out a set of identification experiments to validate the effectiveness of the five feature sets, i. e. the packet size feature set, the first and second order difference feature sets, the statistical feature set, the combine feature set selected by mutual information. We use PS, $Diff1$, $Diff2$, $Statistical$ and MI to represent these feature sets, respectively.

We firstly summarize the identification results in averages in Table 3. And then show detailed accuracy and AUC results in six column plots (Fig. 4 to 9).

When observing the average identification results, it can be found that the packet size feature set and the combine feature set selected by mutual information are the best performed feature sets. PS feature set gets the highest average accuracy for UNIBS data set, and the highest average AUC values for UNIBS and UJN data sets. MI feature set gets the highest average accuracies for Auckland II and UJN data sets, and the highest average AUC value for Auckland

II data set. It should be noticed that all feature sets are able to achieve high identification performances for the three data sets, and the absolute differences among their results are not very significant. Therefore, all these feature sets including the derived and combined feature set are effective for early stage traffic identification.

Fig. 4 and 5 show the accuracy and AUC results for Auckland II data set, respectively. All feature sets get identification accuracies higher than 96%. And all AUC values are greater than 0.98, except the AUC of the $Diff2$ feature set using logistic classifier. It can be seen that for most classifiers, the results of the six feature sets are not significantly different from each other. Especially for the PS feature set and the combined feature set (MI), their Acc and AUC values are always very close.

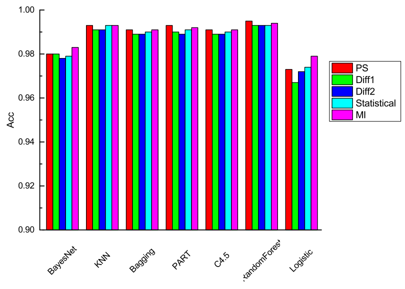


Fig. 4. Acc results of Auckland II data set

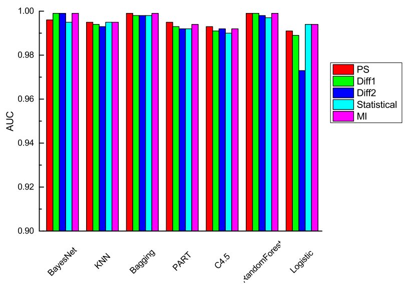


Fig. 5. AUC results of Auckland II data set

The results for UNIBS data set are shown in Fig. 6 and 7. For most classifiers, all the six feature set also get high identification performances as most Acc values are higher than 97% and most AUC values are higher than 0.98. The second difference and the statistical feature sets behave not so well as the others do when using logistic classifier. both of their Acc and AUC values for logistic are obviously lower than that of the other feature sets. However, we can not say that $Diff2$ and statistical feature sets are worse than the others, since the differences

Table 3. Average identification results

Feature sets	Acc			AUC		
	Auckland II	UNIBS	UJN	Auckland II	UNIBS	UJN
PS	0.9880	**0.9879**	0.9406	0.9954	**0.9963**	**0.9331**
Diff1	0.9856	0.9861	0.9397	0.9947	0.9960	0.9180
Diff2	0.9859	0.9810	0.9413	0.9921	0.9939	0.9167
Statistical	0.9871	0.9603	0.9166	0.9944	0.9796	0.9231
MI	**0.9890**	0.9870	**0.9429**	**0.9960**	0.9957	0.9299

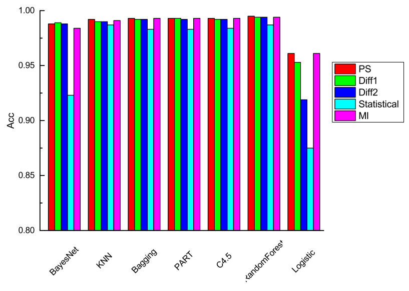


Fig. 6. Acc results of UNIBS data set

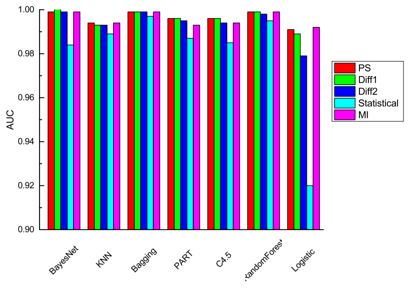


Fig. 7. AUC results of UNIBS data set

of the two feature sets and the other feature sets are not significant for other classifiers.

When observing the results for UJN data set in Fig. 8 and 9, we can see that the differences among the six feature set are more clearer than that of the previous two data sets. PS and MI feature sets behave very similarly. Both Acc and AUC values of the three feature sets are very close for all classifiers. However, the pattern of the other three feature sets are quite different. $Diff1$ and $Diff2$ get Acc values which are significantly higher than that of the other feature set for BayesNet. While their AUC values are close to that of the others. This implies that BayesNet achieves high sensitivities using $Diff1$ and $Diff2$ feature sets, while the specificities are quite low. The statistical featue set does not perform very well for accuracy. However, it performs quite well for AUC. This means that the statistical feature set is able to make a good trade-off between the sensitivity and the specificity. From the point of view of classifiers, logistic does not perform so well as the other classifiers do.

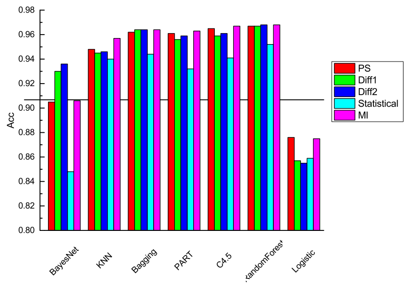


Fig. 8. Acc results of UJN data set

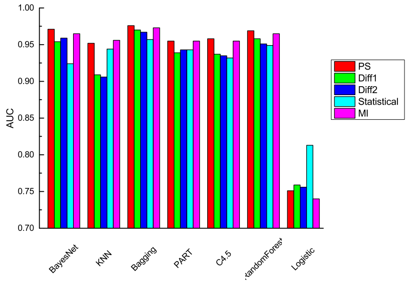


Fig. 9. AUC results of UJN data set

5.3 Analysis and Discussions

According to the experimental results, some lessons can be learned:

- From the global view, the six type of features do not show significant effectiveness differences. Mutual information analysis results have shown that most derived and statistical features gained similar level of mutual information as the packet size feature do. The strongest evidences exist in the identification experimental results. We can hardly tell which feature set outperformed another significantly according to the identification results.
- As the original feature type, the packet size features show their high performances in early stage traffic identification. In most cases, the packet sizes feature set achieved high accuracy and AUC values in our experiments. So, the original packet sizes is effective for early stage identification.
- The combined feature set also performed very well. In this study, we do not focus on the problem of feature selection, and we only build a combined feature set according to the rankings of the mutual information values. Identification results validated the high performances of the combined feature set. Therefore, we infer that more effective feature sets can be built using effective feature selection methods, and this is one topic of our future works.

6 Conclusions

We have tried to evaluate the effectiveness of the packet size based features for early stage traffic identification in this paper. We use the mutual information analysis to find the relationship between features and the traffic type, and also carry out identification experiments to validate the effectiveness of different type of features. Three traffic data sets include two opening data sets and 7 well-known classifiers are applied. According to the experimental results, we can draw some conclusions. Firstly, derived features such as the difference and the statistical features are not more effective than original packet size features. Secondly, the combined features selected by mutual information analysis show high performances in most cases. Although the combined feature set does not outperform the other feature sets significantly, we infer that more effective feature selection methods can pick out combined feature sets that are more effective than single type of features. And this is an important future work of us.

Acknowledgment. This research was partially supported by the National Basic Research Program of China (973 Program) under grant No. 2011CB302605, the National High Technology Research and Development Program of China (863 Program) under grant No. 2012AA012502 and 2012AA012506, the National Key Technology R&D Program of China under grant No. 2012BAH37B01, National Natural Science Foundation of China under grant No. 61173078, No. 61203105, No. 61173079, and No. 61373054, the Provincial Natural Science Foundation of Shandong under grant No. ZR2012FM010, No. ZR2011FZ001, and No.ZR2012FQ016.

References

1. Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., Salamatian, K.: Traffic Classification On The Fly. In: ACM SIGCOMM 2006, pp. 23–26 (2006)
2. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30, 1145–1159 (1997)
3. Dainotti, A., Pescapé, A., Claffy, K.C.: Issues and future directions in traffic classification. IEEE Network 26(1), 35–40 (2012)
4. Dainotti, A., Pescapé, A., Sansone, C.: Early classification of network traffic through multi-classification. In: Domingo-Pascual, J., Shavitt, Y., Uhlig, S. (eds.) TMA 2011. LNCS, vol. 6613, pp. 122–135. Springer, Heidelberg (2011)
5. Estan, C., Varghese, G.: New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. ACM Transactions on Computer Systems 21(3), 270–313 (2003)
6. Este, A., Gringoli, F., Salgarelli, L.: On the Stability of the Information Carried by Traffic Flow Features at the Packet Level. In: ACM SIGCOMM 2009, pp. 13–18 (2009)
7. Este, A., Gringoli, F., Salgarelli, L.: Support Vector Machines for TCP traffic classification. Computer Networks 53, 2476–2490 (2009)
8. Huang, N., Jai, G., Chao, H.: Early identifying application traffic with application characteristics. In: IEEE Int. Conference on Communications (ICC 2008), pp. 5788–5792 (2008)
9. Huang, N., Jai, G., Chao, H., et al.: Application traffic classification at the early stage by characterizing application rounds. Information Sciences 232(20), 130–142 (2013)
10. Huang, J., Ling, C.X.: Using AUC and accuracy in evaluating learning algorithms. IEEE Transactions on Knowledge and Data Engineering 17, 299–310 (2005)
11. Hullár, B., Laki, S., Gyorgy, A.: Early identification of peer-to-peer traffic. In: 2011 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE Press (2011)
12. Gringoli, F., Salgarelli, L., Dusi, M., et al.: Gt: picking up the truth from the ground for internet traffic. ACM SIGCOMM Computer Communication Review 39(5), 12–18 (2009)
13. Li, W., Moore, A.W.: A Machine Learning Approach for Efficient Traffic Classification. In: Proceedings of IEEE MASCOTS 2007, pp. 310–317 (2007)
14. Moore, A.W., Zuev, D., Crogan, M.: Discriminators for use in flow-based classification, Intel Research Tech. Rep. (2005)
15. Moore, A.W., Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. In: ACM SIGMETRICS 2005, pp. 50–60 (2005)
16. Nguyen, T.T.T., Armitage, G., Branch, P., et al.: Timely and continuous machine-learning-based classification for interactive IP traffic. IEEE/ACM Transactions on Networking (TON) 20(6), 1880–1894 (2012)
17. Peng, H.: Mutual infomation Matlab toolbox, `http://www.mathworks.com/matlabcentral/fileexchange/14888-mutual-information-computation`
18. Peng, L., Zhang, H., Yang, B., et al.: Traffic Labeller: Collecting Internet Traffic Samples with Accurate Application Information. China Communications 11(1), 67–78 (2014)
19. Qu, B., Zhang, Z., Guo, L., et al.: On accuracy of early traffic classification. In: IEEE 7th International Conference on Networking, Architecture and Storage (NAS), pp. 348–354. IEEE Press (2012)

20. Tcpdump/Libpcap, `http://www.tcpdump.org`
21. UNIBS: Data sharing, `http://www.ing.unibs.it/ntw/tools/traces/`
22. Waikato Internet Traffic Storage (WITS), `http://www.wand.net.nz/wits`
23. Weka 3: Data Mining Software in Java, `http://www.cs.waikato.ac.nz/ml/weka/`
24. Zhang, J., Xiang, Y., Wang, Y., et al.: Network traffic classification using correlation information. IEEE Transactions on Parallel and Distributed Systems 24(1), 104–117 (2013)

Hyper-Star Graphs: Some Topological Properties and an Optimal Neighbourhood Broadcasting Algorithm

F. Zhang[1], K. Qiu[1], and J.S. Kim[2]

[1] Department of Computer Science
Brock University
St. Catharines, Ontario, L2S 3A1 Canada
[2] Dept. of Computer Science and Technology
University of Rochester
Rochester, NY 14627 USA
U.S.A.

Abstract. Hyper-star graph $HS(2n, n)$ was introduced to be a competitive model to both hypercubes and star graphs. In this paper, we study its properties by giving a closed form solution to the surface area of $HS(2n, n)$ and discussing its Hamiltonicity by establishing an isomorphism between the graph and the well known middle levels problem. We also develop a single-port optimal neighbourhood broadcasting algorithm for $HS(2n, n)$.

Keywords: Interconnection network, hyper-star, surface area, Hamiltonicity, neighbourhood broadcasting.

1 Introduction

Advances in hardware technology, especially the VLSI circuit technology, have made it possible to build a large-scale multiprocessor system that contains thousands or even tens of thousands of processors. One crucial step on designing a large-scale multiprocessor system is to determine the topology of the interconnection network (network for short), because the system performance is significantly affected by the network topology. A network is conveniently represented by a graph whose nodes represent the processors of the network and whose edges represent the communication links of the network. Throughout this paper, we use network and graph, processor and node, and link and edge, interchangeably.

One of the most popular and efficient interconnection networks is that of hypercubes [14,20]. Another family of regular graphs, star graphs [1,2], has been extensively studied. Hyper-star graphs $HS(m, n)$ and folded hyper-star graphs $FHS(2n, n)$ were introduced by Lee et al. [19] to be new alternatives to both hypercubes and star graphs. The hyper-star graph is a regular network, when $m = 2n$. The hyper-star has many attractive properties. For examples, it has better scalability, simple routing algorithm, maximum fault-tolerance, and lower network cost (defined as the product of its degree and diameter) than the hypercube and its variations [19]. Many properties of hyper-star and folded hyper-star graphs were introduced in [6,9,15,16,17,18,21,31]. Please note that there is a similarly named network called hyperstar [3] which is completely different from the hyper-star studied in this paper.

In this paper, we study additional properties of the hyper-star such as the surface area and its Hamiltonicity. We also develop an optimal algorithm for neighbourhood broadcasting on $HS(2n, n)$. Specifically, after we introduce the hyper-star graph in Section 2, we show that the surface area of $HS(2n, n)$ is $\binom{n}{\lceil \frac{k}{2} \rceil}\binom{n-1}{\lfloor \frac{k}{2} \rfloor}$ in Section 3. We then show in Section 4 that $HS(2n, n)$ is isomorphic to the well known middle cube, thus linking the Hamiltonicity of $HS(2n, n)$ to that of the middle cube, which remains an open problem. Finally, in Section 5, we find an optimal algorithm for performing neighbourhood broadcasting on $HS(2n, n)$.

2 Preliminaries

Both the hypercube and the star are popular topologies to interconnect many processors in a parallel computer. In a hypercube Q_n of dimension n, or n-cube for short, there are 2^n nodes where nodes i and j are connected if if their binary representations differ on one bit. For example, in a 5-cube, node 01011 is connected to 11011, 00011, 01111, 01001, and 01010. In a star graph S_n of dimension n, there are $n!$ nodes where each node is a permutation of symbols 1, 2, ..., n. Two permutations are connected if one can be obtained from the other by swapping its symbols at positions 1 and i, $2 \leq i \leq n$. For example, in a 5-star, node 31452 is connected to 13452, 41352, 51432, and 21453. The star graph is an attractive alternative to the hypercube, and compares favorably with it in several aspects [1,2]. For example, both the degree and diameter of S_n are $O(n)$, i.e., sub-logarithmic in the number of vertices of S_n, while a hypercube with $O(n!)$ vertices has a degree and diameter of $O(\log n!) = O(n \log n)$, i.e., logarithmic in the number of vertices. Also, it is known that the star graph is both vertex symmetric and edge symmetric. The hyper-star attempts to combine the advantages of both networks.

A *hyper-star graph* $HS(m, n)$ is an undirected graph consisting of $\binom{m}{n}$ nodes. For a node u whose binary representation is $u = s_1 s_2 \cdots s_m$, $u \in HS(m, n)$ if the Hamming weight of u, $H(u) = n$. Two nodes are adjacent if and only if one can be obtained from the other by exchanging the first symbol with a different symbol (1 with 0, or 0 with 1) in another position. Two nodes directly connected by an edge are said to be *neighbours*. Every node in $HS(m, n)$ has degree n or $m - n$. Let $dist(u, v)$ be the distance from $u = u_1 u_2 \ldots u_{2n}$ to $v = v_1 v_2 \ldots v_{2n}$ in $HS(2n, n)$. If a bit string $R = r_1 r_2 \cdots r_{2n}$ is obtained by applying the bitwise Exclusive-OR operation to u and v, i.e., $r_i = u_i \oplus v_i$, then $dist(u, v) = \sum_{i=2}^{2n} r_i$. Thus, the diameter of $HS(2n, n)$ is $2n - 1$. When $m = 2n$, the hyper-star graph is a regular graph $HS(2n, n)$, where a node is represented by a string of $2n$ bits with n of them being 1. We write a node $\overbrace{0\ldots0}^{n}\overbrace{1\ldots1}^{n}$ in $HS(2n,n)$ as $e_n = 0^n1^n$. we call e_n the identity node in $HS(2n,n)$. Fig. 1 shows $HS(6,3)$.

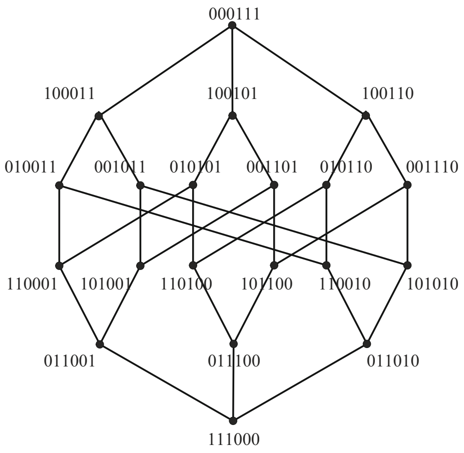


Fig. 1. $HS(6,3)$

Lemma 1 *[21] Let $u = 0^n1^n$ and v be two nodes in $HS(2n,n)$, P be the shortest path with a length ρ (≥ 3) from u to v and Q be a path that is some ith cyclically permuted version of P. There are no common internal nodes on P and Q.*

Lemma 2 *[9] The length of the shortest cycle in $HS(2n,n)$ is 6.*

3 Surface Area of $HS(2n,n)$

Given a node u in a graph G, a question one may ask is how many nodes are at distance k from u, $k \in [0, D(G)]$, where $D(G)$ is the diameter of G. This quantity, denoted as $B_{G,u}(k)$, is referred to, in the literature, as the "Whitney numbers of the second kind of the poset" [23], the surface area of a vertex with radius k [13], or the "distance distribution" of nodes in a graph [29]. This surface area problem has been studied for a variety of graphs [7,8,13,23,29].

The surface area of a graph can find several applications in network performance evaluation, e.g., in computing various bounds for the problem of k-neighbourhood broadcasting [11] in interconnection networks, and in deriving the "transmission of a graph" [25], defined as the sum of all the distances in a graph G, an indicator for the speed of an average communication. This notion of transmission is also suggested to achieve the generalized Moore bound, an important concept in extremal graph theory.

This quantity of surface area is especially well defined for the node symmetric graphs, as the surface area for any node in a node symmetric graph G equals that for any other node in G. Since $HS(2n,n)$ is node-symmetric [6,15], we focus on the surface area of one specific node, $e_n = 0^n1^n$.

Definition 1 *Let σ_i be an operation that exchanges s_1 and s_i, $2 \leq i \leq 2n$, where s_i is the complement of s_1. Then two nodes u and v are connected when v is obtained from the operation $\sigma_i(u)$, $2 \leq i \leq 2n$.*

Definition 2 *For a node u, we denote by $[k_1, k_2, \ldots, k_t]$ a path obtained by applying operations $\sigma_{k_1}, \sigma_{k_2}, \ldots, \sigma_{k_t}$ in sequence from the node u.*

For example, there is a path [3, 2, 4] or [4, 2, 3] from 0011 to 1100. Since a shortest path from e_n can be constructed by applying unique operations σ_i, $n+1 \leq i \leq 2n$, and σ_j , $2 \leq j \leq n$, alternately, we have the following lemma.

Lemma 3 *Any shortest path $[k_1, k_2, \ldots, k_t]$ from the node $e_n = 0^n1^n$ to a specific node in $HS(2n,n)$ has the same set of numbers $\{k_1, k_3, \ldots, k_i, \ldots\}$ (i is odd) and $\{k_2, k_4, \ldots, k_j, \ldots\}$ (j is even).*

We refer to the set $\{k_1, k_3, \ldots, k_i, \ldots\}$ as S_{od}, and the set $\{k_2, k_4, \ldots, k_j, \ldots\}$ as S_{ev}. Notice here that every number in S_{od} is between $n+1$ and $2n$, and every number in set S_{ev} is between 2 and n. Thus we can see that the number of nodes of a certain distance from e_n is determined by the number of different combinations of S_{od} and S_{ev}. This leads us to the following theorem.

Theorem 1 *The surface area of $HS(2n,n)$, for $1 \leq k \leq D(HS(2n,n) = 2n-1$, is*

$$B_{HS(2n,n),e_n}(k) = \binom{n}{\lceil \frac{k}{2} \rceil}\binom{n-1}{\lfloor \frac{k}{2} \rfloor}. \qquad (A)$$

Table 1 shows the surface area of $HS(2n,n)$, $2 \leq n \leq 6$.

Table 1. Surface Area of $HS(2n,n)$

Radius k	$HS(4,2)$	$HS(6,3)$	$HS(8,4)$	$HS(10,5)$	$HS(12,6)$
1	2	3	4	5	6
2	2	6	12	20	30
3	1	6	18	40	75
4	-	3	18	60	150
5	-	1	12	60	200
6	-	-	4	40	200
7	-	-	1	20	150
8	-	-	-	5	75
9	-	-	-	1	30
10	-	-	-	-	6
11	-	-	-	-	1

Interestingly, the sequence 2, 2, 1, 3, 6, 6, 3, 1, 4, 12, 18, 18, 12, 4, 1, ... appears in the On-line Encyclopedia of Integer Sequences (OEIS) [28] as sequence A088459 and is also a part of Dyck paths of semi-length n with k peaks (OEIS A088855) [4]. The function is

$$\binom{\lfloor \frac{n}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}\binom{\lceil \frac{n}{2} \rceil}{\lceil \frac{k}{2} \rceil}. \qquad \text{(B)}$$

$$\binom{\lfloor \frac{2n-1}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}\binom{\lceil \frac{2n-1}{2} \rceil}{\lceil \frac{k}{2} \rceil}. \qquad \text{(C)}$$

(C) is the part of (B), and (A)=(C).

4 Hamiltonicity of $HS(2n, n)$

The Hamiltonicity problem is yet another important problem often studied for interconnection networks.

Definition 3 *Suppose that G is an interconnection network. A path (or cycle) in G is called a Hamiltonian path (or Hamiltonian cycle) if it contains every node of G exactly once. G is called Hamiltonian if there is a Hamiltonian cycle in G.*

The interconnection network middle cube [26,27] is defined as:

Definition 4 *Let Q_n be the n-dimensional hypercube. If $n = 2d + 1$, then the subgraph M_n of Q_n induced by the nodes having exactly d or $d + 1$ 1's is called the middle cube of dimension n.*

The middle cube is first studied as a potential interconnection network for parallel computation in 1990 [22]. There is one well-known conjecture concerning the middle cube, namely the *Revolving Door (Middle Levels) Conjecture* [30], which is stated as follows:

Conjecture 1 *All middle cubes M_n, $n > 1$, are Hamiltonian.*

This conjecture has been verified/proved for $n \leq 35$ [27], but is still open in general.

In the course of trying to solve the Hamiltonicity problem of the hyper-star graph, we try to establish a relationship between the hyper-star and the middle cube due to the resemblance between their representation strings. And this leads us to the following theorem:

Theorem 2 *The hyper-star graph $HS(2d+2, d+1)$ is isomorphic to the middle cube M_{2d+1}, $d \geq 0$.*

Proof. Define a mapping $\phi : V(M_{2d+1}) \longrightarrow V(HS(2d + 2, d + 1))$ as follows. If α is a node of M_{2d+1}, and the Hamming weight of α is $d + 1$, then define $\phi(\alpha) = 0\alpha$; If β is a node of M_{2d+1}, and the Hamming weight of β is d, then define $\phi(\beta) = 0\beta$. Now consider α, a node of M_{2d+1}, $H(\alpha) = d + 1$. Then the neighbours of α must be in the form of β, $H(\beta) = d$. And α and β differ in exactly one bit, say bit position p. Furthermore, bit p of α is 1, bit p of β is 0. Then, 0α and 1β differ in two bits, the first bit of course, and the bit at position $p+1$. The bit at position $p+1$ is 1 in 0α and is 0 in 1β. This means 1β can be obtained from 0α by exchanging the first bit with the $(p+1)$th bit, which implies $\phi(\alpha)$ and $\phi(\beta)$ are two adjacent nodes in the hyper-star graph $HS(2d+2, d+1)$. Hence the proof is complete. □

By this theorem, we claim that the Hamiltonicity problem of $HS(2n, n)$ is as hard as the Hamiltonicity problem of the middle cube graph, and hence the following conjecture:

Conjecture 2 *All regular hyper-star graphs $HS(2n, n)$, $n \geq 1$, are Hamiltonian.*

5 Neighbourhood Broadcasting of $HS(2n, n)$

The neighbourhood broadcasting problem (NBP, for short) is the problem of disseminating a message from the source node to all the nodes adjacent to the source node [5,10,12,24]. The neighbourhood broadcasting problem was first introduced in [10] as a tool to simulate a single step of the all-port model by the single-port model in a given network. Since then, it has been extended to k-neighbourhood broadcasting where a node u is to send information to all nodes at distances less than or equal to k to u [11]. On the all-port model, a node can communicate with all of its neighbours at the same time. In this case, NBP becomes a trivial problem and takes one time unit. From now on, we consider NBP on single-port model. On the single-port model, a node can only communicate with one of its neighbours at a time. A trivial lower bound of NBP can be stated as follows: Any neighbourhood broadcasting algorithm on a network with degree d must require $\Omega(\log d)$ time. This is because at each time unit, one processor with the message can only send it to one of its neighbours, so after every step, the number of neighbours which have received the message can at most double. The maximum number of neighbours of a node in the network is d, thus the least time needed to solve NBP is $\Omega(\log d)$.

$HS(2n, n)$ has a degree of n. Thus the lower bound for NBP on $HS(2n, n)$ is $\Omega(\log n)$. Also, $HS(2n, n)$ is node-symmetric [6,15]. Without loss of generality, we assume the source node for NBP is $e_n = 0^n1^n$. By Lemma 1 and 2, we have the following.

Theorem 3 *Any pair of neighbours of the source node $e_n = 0^n1^n$ is connected by a path of length 4, and these paths between different pairs of neighbours are node-disjoint.*

Theorem 3 allows us to view the source node e_n together with its n neighbours as a de facto complete graph in the sense that any two nodes are connected by a path of constant length. Based on the technique of recursive doubling where at each step, we double the number of neighbours with the message by using a set of node-disjoint paths with constant length between neighbours, resulting in a simple and optimal neighbourhood broadcasting algorithm for $HS(2n,n)$. We denote the neighbour obtained by switching the first bit of the source node e_n with the $(n+i)$th bit, $1 \leq i \leq n$, the ith neighbour of the source node e_n.

Table 2. Simple neighbourhood broadcasting algorithm for $HS(2n,n)$

- For the first step, the source node e_n sends the message to its first neighbour via direct link.
- At step i, $i \geq 2$, the source node e_n sends the message to its 2^{i-1}th neighbour. If neighbour j has the message, then at this step, it sends the message to neighbour $j + 2^{i-1}$ via the 4-length path $[2, j+2^{i-1}, j, 2]$.
- The process continues until all the neighbours of the source node e_n have the message.

Since the number of nodes with the message is doubled after each step (except possibly after the last step), the number of steps required is $O(\log n)$. Each step takes constant time because the cycles used have a fixed length of four. So in view of the $\Omega(\log n)$ lower bound, our algorithm is asymptotically optimal.

For example, in $HS(14,7)$, the neighbourhood broadcasting is done in the following Steps. $\Rightarrow$ and t mean message transfer and tth neighbour, respectively $(1 \leq t \leq 7)$.

- Step 1: $(e_n \Rightarrow 1)$ 00000001111111 $\Rightarrow$ 10000000111111.
- Step 2: $(e_n \Rightarrow 2)$ 00000001111111 $\Rightarrow$ 10000001011111.
  $(1 \Rightarrow 3)$ 10000000111111 $\Rightarrow$ 01000000111111 $\Rightarrow$ 11000000101111 $\Rightarrow$ 01000001101111 $\Rightarrow$ 10000001101111.
- Step 3: $(e_n \Rightarrow 4)$ 00000001111111 $\Rightarrow$ 10000001110111.
  $(1 \Rightarrow 5)$ 10000000111111 $\Rightarrow$ 01000000111111 $\Rightarrow$ 11000000111011 $\Rightarrow$ 01000001111011 $\Rightarrow$ 10000001111011.
  $(2 \Rightarrow 6)$ 10000001011111 $\Rightarrow$ 01000001011111 $\Rightarrow$ 11000001011101 $\Rightarrow$ 01000001111101 $\Rightarrow$ 10000001111101.
  $(3 \Rightarrow 7)$ 10000001101111 $\Rightarrow$ 01000001101111 $\Rightarrow$ 11000001101110 $\Rightarrow$ 01000001111110 $\Rightarrow$ 10000001111110.

6 Conclusion

In this paper, we presented some properties such as the surface area and Hamiltonicity of $HS(2n,n)$. we also found an optimal neighbourhood broadcasting for the network. First, we showed that the surface area of $HS(2n,n)$ is $\binom{n}{\lceil \frac{k}{2} \rceil}\binom{n-1}{\lfloor \frac{k}{2} \rfloor}$. We then conjectured that regular hyper-star graph $HS(2n,n)$, $n \geq 1$, is Hamiltonian by establishing an isomorphism between $HS(2n,n)$ and the middle cube. In addition, we presented a simple and optimal neighbourhood broadcasting algorithm of $HS(2n,n)$. We hope to find more algorithms that can run on this interconnection network in the future.

References

1. Akers, S.B., Harel, D., Krishnamurthy, B.: The Star Graph: an Attractive Alternative to the n-Cube. In: Proceedings of the International Conference on Parallel Processing, pp. 393–400 (1987)
2. Akers, S.B., Krishnamurthy, B.: A Group Theoretic Model for Symmetric Interconnection Networks. IEEE Trans. on Compu. c-38 (4), 555–566 (1989)
3. Al-Ayyoub, A.-E., Day, K.: The hyperstar Interconnection Network. Journal of Parallel and Distributed Computing 48, 175–199 (1998)
4. Barry, P.: On Integer-Sequence-Based Constructions of Generalized Pascal Triangles. Journal of Integer Sequences 9, 1-34 (2006)
5. Bermond, J.C., Ferreira, A., Perennes, S., Peters, J.G.: Neighbourhood Broadcasting in Hypercubes. SIAM Journal on Discrete Mathematics 21(4), 823–843 (2007)
6. Cheng, E., Lipták, L.: Structural Properties of Hyper-stars. Ars Combinatoria 80, 65–73 (2006)
7. Cheng, E., Qiu, K., Shen, Z.: On the Surface Areas and Average Distances of Meshes and Tori. Parallel Processing Letters 21(1), 61–75 (2011)
8. Cheng, E., Qiu, K., Shen, Z.: On the Surface Area of the Augmented Cubes. Journal of Supercomputing 61, 856–868 (2012)
9. Cheng, E., Shah, M.: A Strong Structural Theorem for Hyper-stars. Congressus Numerantium 179, 181–191 (2006)
10. Cosnard, M., Ferreira, A.: On the Real Power of Loosely Coupled Parallel Architectures. Parallel Processing Letters 1(2), 103–111 (1991)
11. Fertin, G., Raspaud, A.: *k*-Neighbourhood Broadcasting. In: Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2001), pp. 133–146 (2001)
12. Fujita, S.: Optimal neighborhood Broadcast in Star Graphs. Journal of Interconnection Networks 4(4), 419–428 (2003)
13. Imani, N., Sarbazi-Azad, H., Akl, S.G.: Some Topological Properties of Star Graphs: the Surface Area and Volume. Discrete Mathematics 309(3), 560–569 (2009)
14. Johnson, S.L., Ho, C.T.: Optimal Broadcasting and Personalized Communication in Hypercubes. IEEE Trans. Computers 38(9), 1249–1268 (1989)
15. Kim, J.-S., Oh, E., Lee, H.-O., Heo, Y.-N.: Topological and Communication Aspects of Hyper-Star Graphs. In: Yazıcı, A., Şener, C. (eds.) ISCIS 2003. LNCS, vol. 2869, pp. 51–58. Springer, Heidelberg (2003)
16. Kim, J.S., Cheng, E., Lipták, L., Lee, H.O.: Embedding Hypercubes, Rings and Odd Graphs into Hyper-stars. International Journal of Computer Mathematics 86(5), 771–778 (2009)
17. Kim, J.S., Cheng, E., Lipták, L., Lee, H.O.: A Note on Embedding among Folded Hypercubes, Even Graphs and Odd Graphs. International Journal of Computer Mathematics 8(5), 882–891 (2011)
18. Kim, J.S., Kim, S.W., Cheng, E., Lipták, L.: Topological Properties of Folded Hyper-star Networks. Journal of Supercomputing 59, 1336–1347 (2012)
19. Lee, H.O., Kim, J.S., Oh, E., Lim, H.S.: Hyper-Star Graph: A New Interconnection Network Improving the Network Cost of the Hypercube. In: Shafazand, H., Tjoa, A.M. (eds.) EurAsia-ICT 2002. LNCS, vol. 2510, pp. 858–865. Springer, Heidelberg (2002)
20. Leighton, T.: Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufman, San Mateo (1992)

21. Lipták, L., Cheng, E., Kim, J.S., Kim, S.W.: One-to-Many Node Disjoint Paths of Hyper-Star Networks. Discrete Applied Mathematics 160, 2006–2014 (2012)
22. Madabhushi, S., Lakshmivarahan, S., Dhall, S.: Analysis of the Modified Even Networks. Technical Report, School of Electrical Engineering and Computer Science, University of Oklahoma (1990)
23. Portier, F., Vaughan, T.: Whitney Numbers of the Second Kind for the Star Poset. European Journal of Combinatorics 11(3), 277–288 (1990)
24. Qiu, K.: On a Unified Neighbourhood Broadcasting Scheme for Interconnection Networks. Parallel Processing Letters 17(4), 425–437 (2007)
25. Sampels, M.: Vertex-Symmetric Generalized Moore graphs. Discrete Applied Mathematics 138, 195–202 (2004)
26. Savage, C.D., Winkler, P.: Monotone Gray code and the Middle Levels Problem. Journal of Combinatorial Theory, Series A 70, 230–248 (1995)
27. Shields, I., Shields, B.J., Savage, C.D.: An Update on the Middle Levels Problem. Discrete Mathamatics 309, 5271–5277 (2009)
28. Sloane, N.J.A.: The On-Line Encyclopedia of Integer Sequences, `http://oeis.orgo`
29. Wang, L., Subramanian, S., Latifi, S., Srimani, P.K.: Distance Distribution of Nodes in Star Graphs. Applied Mathematics Letters 19(8), 780–784 (2006)
30. West, D.B.: Revolving Door (Middle Levels) Conjecture, `http://www.math.uiuc.edu/~west/openp/revolving`
31. Yang, J.S., Chang, J.M.: Independent Spanning Trees on Folded Hyper-stars. Networks 56(4), 272–281 (2010)

Customized Network-on-Chip for Message Reduction

Hongwei Wang[1], Siyu Lu[1], Youhui Zhang[1,2,*], Guangwen Yang[1], and Weimin Zheng[1,2]

[1] Department of Computer Science and Technology, Tsinghua University Beijing, China
[2] Technology Innovation Center at Yinzhou Yangtze Delta Region Institute of Tsinghua University, ZheJiang
zyh02@tsinghua.edu.cn

Abstract. This paper proposes a network-on-chip (NoC) design customized for message reduction, which enhances some common routers with a special Reduce Processing Unit (RPU) to complete reduce-computations hop-by-hop, as well as to learn the transmission path of reduction-messages adaptively. More specifically, for reduction on a small data-set, the corresponding data is transmitted through the NoC directly. Thus, along the transmission path, enhanced routers can complete reduction in place, which not only speeds up the processing procedure but also coalesces messages. An adaptive method for the deterministic routing algorithm is also introduced to enable these routers to learn transmission paths accurately to improve the processing efficiency. We present the detailed micro-architecture design and evaluate the corresponding performance, the power consumption and chip-area. Testing results show that this design can improve the *reduction* / *all_reduce* performance of 2.67~11.76 times, while the consumption of power and chip-area are both limited.

Keywords: Network on chip, CMP, message reduction.

1 Introduction

The general trend in processor development has moved from dual- and quad-core processor chips to ones with tens or even hundreds of cores [1][2] that are connected by Network-on-Chip (NoC). In addition, future multi-core-processors (CMPs) are expected to have an amount of local on-chip memory assigned to each core [3]. Then, the architectural similarities of such NoC-based CMPs and computer clusters have led to the adoption of the message passing mechanism for on-chip programming, like RCCE for Intel's SCC [4] or the Multicore Communication API [5].

MPI is a standardized and portable message-passing system designed to function on a wide variety of parallel computers. It provides a large amount of MPI collective operations [6]. MPI_Reduce and MPI_Allreduce are widely used because of the simplification of the complex task of writing scalable, parallel programs [7]. [8] showed

[*] Corresponding author.

that the execution time of such operations can account for up to 40% of the total execution time of MPI routines. Accordingly, we could achieve comparatively high performance by improving the implementation of message-reduction.

A naive implementation of reduction is based on point-to-point primitives [9]. When a reduction-operation is initiated, all nodes send data to the *rank root* node. The latter receives the data and handles all the computation. To improve efficiency, two types of optimization can be employed: (1) To parallelize computation and communication. (2) To offload overheads to some specific hardware module(s).

Software-based solutions usually belong to the first category. Binomial Tree Reduce (BTR)[10] is an algorithm that takes $\log_2 P$ (P is the number of processors) steps to get the final result. This is one of the most efficient algorithms by software. The essential idea is: The associativity of computation allows parallel reduction. Each level of the tree corresponds to one round of the reduction-algorithm. From the transferring aspect, it can be regarded as in-transmission optimization: the original and intermediary reduction-operands are transmitted across the given communication paths and computed by intermediary nodes, till the final result has been gotten by the root. However, the application-level topology may not be corresponding to actual physical positions, which may lead to extra overheads.

The second category, hardware-assistant specific technique, is widely used to minimize software overheads. Sun Clint network[12], Voltaire FCA[13], Intelligent Network Interface Card (INIC)[14], BlueGene/L[15], Reconfigurable Hardware on Accelerating MPI_Reduce [16] and so on are such solutions in the high-performance computing or reconfiguration computing fields. Some works have provided hardware supports in the CMP chips, like [17] and the TILE processor [18]. Some others have implemented software optimizations based on specific hardware features, like [19] [20] on Cell, [21] on SCC, etc.

Some work has used the two types of optimization together. For example, the SCC work [21] has completed the reduction step by step on the on-chip MPBs (message-passing buffers) along the transmission paths, while the message processing is still achieved by software; [16] on FPGA also has adopted some specific computation patterns for the optimization.

Compared with existing work, we propose a network-on-chip (NoC) design customized for message reduction, which enhances common NoC routers to complete reduction hop-by-hop. The main characteristic lies in that all enhancements are completely integrated into the NoC-router's architecture. Namely, the in-transmission optimization is united with hardware-offloading on the network layer.

Considering the CMP architecture connected by a NoC, message-packets generated by reduction-calls can be transmitted through the NoC directly: they are delivered between nodes, across intermediate routers from the source node to the destination, namely the hop-by-hop transmission. Under this condition, a node may act as an intermediate communication node for more than one packet. This kind of transmission gives us an opportunity to do some optimization along the transmission path, such as computation and coalescing messages.

Accordingly, we have enhanced routers so that they can not only complete reduction-computation but also learn the transmission paths of reduction-messages adaptively. Namely, the computation and communication have been combined together. Unlike the current in-transmission optimization that has to construct some communication pattern in advance, our work can adapt to the NoC context inherently because any message-packet of a reduction (we call it "*reduction packet*" in this paper for short) will be transmitted by the NoC after all.

To the best of our knowledge, such approach on NoC has not received much attention yet. One similar research is [22]: it has presented a method to provide network-hardware support for broadcast and reduction-operations. But this work is implemented in FPGAs and limited to the specific FPGA interconnection architecture (BEE3 platform) and topology.

With this design, the following contributions have been accomplished :

1. We enhance a common NoC router by integrating a processing unit into the router pipeline. So that along the transmission paths of *reduction packet*s, reduction can be completed by enhanced routers on the network layer.
2. A SW/HW hybrid method is proposed to enable routers to learn transmission paths adaptively. Thus a router on the path can know the number of *reduction packets* it should process, as well as the direction of each packet, which improves the processing efficiency further.
3. We present an optimal layout of enhanced routers in the NoC, which can make a balance between the performance enhancement and extra overhead. Testing results show that this design can promote the reduce / *all_reduce* performance of 2.67~11.76 times (up to 11.76 for reduction and 10.2 for *all_reduce*), while the consumption of power and chip-area are both very limited.

2 Related Work

2.1 Software Approaches

[10] uses Binomial Tree Reduce (BTR) algorithm for reduce. It follows a special traffic pattern, called Recursive Distance Doubling (RDD), which also forms the basis for many collective operations.

The concept of Recursive Vector Halving is to splitting the input vectors in half each round. Based on this mechanism, for long messages, [11] introduced an approach to compute different portions of the result in parallel on all cores.

2.2 Hardware Approach

Hardware-assistant specific technique is widely used to enhance MPI implementation and minimize software overhead.

In the high-performance computing filed, an FPGA-based implementation of SunMPI-2 APIs for the Sun Clint network is reported by [12]; the Voltaire FCA [13] goes further to off-load collective communication and operations to the network by adding CPUs to the switches. Similarly, [14] implemented MPI Reduce in the FPGA fabric of a Network Interface Card. BlueGene/L [15] have a dedicated network to handle collective communications, specifically for broadcast and reduction. In [16], the semantics of the MPI_Reduce call have been implemented in the reconfigurable resources of an FPGA device across a cluster of all-FPGA compute nodes. A recent work on CMP is [17]: It presents an underlying customized NoC that incorporates buses into NoC to achieve high performance for both point-to-point and broadcast data transmission; an MPI engine attached to each core has implemented basic MPI primitives to relieve the processor core (but no reduce-specific support). In addition, the famous TILE processor [18] supports passing messages between cores without system software intervention.

One similar work is [22]. [22] has enhance the NoC in FPGA make it MPI aware by adding hardware support for broadcast and reduction. The major difference lies in that this work is limited to the specific FPGA interconnection architecture and topology, and then is not so efficient as our work that enables routers to learn transmission paths adaptively.

3 Proposed Architecture

This section describes the proposed design for accelerating reduction with the NoC-level support. There are two kinds of enhanced routers in this work, one of them is referred as Class_1 router. A Class_1 router is a common router integrated with a specific Reduce Processing Unit (RPU); the latter is responsible for computing and coalescing *reduction packets* as well as learning transmission paths. The other is referred as Class_2, integrated with a simplified RPU that is only capable of learning transmission paths.

3.1 Architecture Overview

Before the detailed design, we outline the basic architecture of the objective CMP and some message-passing primitives.

There are N*N computing nodes connected by a 2D-mesh NoC; each node is composed of a CPU core with the local cache and an enhanced router (Class_1 router or a Class_2). There are 5 ports of a router, four of them are used to connect to each of its neighbor routers through independent network channels (North, East, South, and West) and the last one is used to connect to the local core. A deterministic routing algorithm (Y-X routing) has been applied, as well as the virtual-channel flow control.

Without loss of generality, the L1 cache-line is set to 64–byte and we limit the maximum payload-size of a *reduction packet* to one line. It means that a reduction on some small data-set (the result size is not large than 64-byte) is directly supported by our design. In addition, the NoC flit size is set to 128-bit and five flits constructs such a maximum packet.

From the message passing aspect, a core will send out an MPI packet (or more accurately, send out in-order flits of the packet) to the local router; the router will deliver flits to the destination hop by hop. For a common NoC, routers are MPI-unaware, which means they just deliver packets while any other work is handled by upper-level modules. In contrary, our design can complete reduction in the network layer and the details are presented in the following paragraphs.

3.2 The Learning Method

We propose a SW/HW hybrid method to enable routers to learn the transmission path of *reduction packets* adaptively under deterministic routing. More specifically, this method enables every enhanced router to know the number of *reduction packets* it should process, as well as the incoming direction of each packet.

In our design, as a parallel MPI-task is beginning, the runtime will launch a *fake-reduction* on the default MPI communicator: all involved cores (here we assume that one core just execute one MPI process) send a special *reduction packet* without any data payload to the *rank root* node. During the transmission, if a Class_1 router receives such a packet from a neighbor or the local core, it will increase the number of received packets from the corresponding direction and record this value into an internal bit-array, called a learned status-bit-array (abbreviated as LSBA).

Figure 1 shows an example of a Class_1's LSBA. It contains 5 fields; each field records the number of received *reduction packets* from the corresponding direction. It can be inferred from this figure that the router have 3 *reduction packets* from the west neighbor and 5 *reduction packets* from the north, as well as 1 from the local core. Obviously, for an NxN mesh NoC, the length of each LSBA-field is not more than Log_2 (NxN) bits. The exception is the local field: it is only one-bit long because we assume no more than one *reduction packet* of a given communicator will be delivered from the local core. If such a case happens (for example, one core runs two or more processes of a single MPI task), we assume that the runtime software can coalesces messages by completing the reduction locally.

East	South	West	North	Local Core
0 0 0 0	0 0 0 0	0 0 1 1	0 1 0 1	1

Fig. 1. An example of LSBA

If a Class_2 router receives such a packet, it just records the incoming direction, not the number. Thus, the bit-length of a Class_2's LSBA is only 5.

Another issue is about how to forward the empty *reduction packet*. It also depends on which class the router belongs to: (1) If it is a Class_1 router, which means it completes reduction of all incoming *reduction packets* in place and send out one result packet, the router will only forward the first *reduction packet* and discard others (if existing). (2) If Class_2, the packet will be forwarded as normal.

During the common reduction procedure, a Class_1 router will record the information of received *reduction packets* into another bit-array, called a current status-bit-array (abbreviated as CSBA). The structure of a CSBA is the same as LSBA. As the values of these two bit-arrays are identical (it means the router has received all packets that it should handle because we use the deterministic routing algorithm), it can complete the reduction and send out a new *reduction packet* containing the result.

Moreover, every time a MPI communicator is being created or modified, a *fake-reduction* will be carried out again to reflect the latest transmission-path-information of the given communicator. Accordingly, one router with *m* LSBA-structures can support *m* communicators at the same time; each LSBA is identified by the communicator.

For CSBA, the case is a little more complicate because one communicator may launch more than one reduction simultaneously. Therefore, we should differentiate them. The solutions is straightforward: the runtime will allocate an ID for each reduction, which contains the value of the corresponding communicator and a unique number with a fixed length. Thus, the router can use this ID to locate the corresponding CSBA and LSBA.

Another function of the learning method is to speed up the broadcast of *all_reduce* in the NoC layer. From the transmission aspect, the broadcast is an inverse procedure of reduction. As the *rank root* node has gotten the final reduction result, it will send out the result packet(s) according to the LSBA information. It means that if some field of the LSBA is non-zero, the router will transmit one result packet to the corresponding direction. This step will be repeated by each router (regardless of Class_1 or 2) along the transmission path and then all involved nodes will get the result finally.

3.3 Architecture of the Enhanced Router and the Packet Format

The micro-architecture of an enhanced router is shown in Figure 2. A Class_1 router is integrated with a Reduce Processing Unit (RPU) that is responsible for doing the reduction-computation (including MAX, AND, ADD, etc.) on payloads of incoming *reduction packets*, leading to a high efficient mechanism that makes messages being processed during their transmission. While a Class_2 router is integrated with a simplified RPU (sRPU) that doesn't contain FPU, and has less status-bit-arrays.

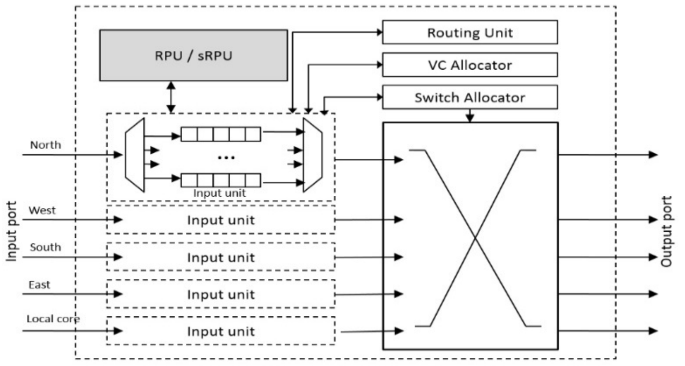


Fig. 2. Block diagram of enhanced router architecture

In addition, the specific format of a *reduction packet* has been introduced, which contains extra information in its head flit, including the reduction type, tag, data type, etc. The head-flit format of a *reduction packet* is shown in Figure 3, which provides information such as Src as the source ID in the network layer, as well as Drc as the destination ID. Com is used to identify different communicators and reductions of one communicator. Op type refers to the corresponding reduction type. If all bits of Op are 0, it means that this packet is a special *reduction packet* for the learning step.

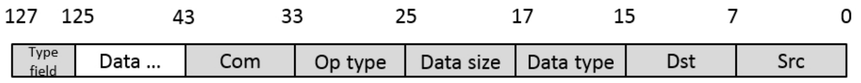


Fig. 3. Head-flit format of *Reduction packet*

The original input unit has been modified accordingly: It can judge whether an incoming flit belongs to a *reduction packet* or not. If yes, it will piece together all such flits into the virtual-channel flit-buffers.

- RPU

Figure 4 shows the block diagram of the RPU, which contains the following main modules:

– A control unit: It is in charge of the reduction management; it also contains some registers to store the reduction type, tag, data type, etc.
– A simplified Float Point Unit (FPU), which can complete reduction of two double-precision floating points once, including MAX, MIN, ADD, MULTIPLY, etc. Of course it can process integer data, too. Furthermore, the division function is needless because it does not meet with the associativity.
– Three FIFOs. Two FIFOs (FIFO_A and FIFO_B) are used as the double buffer for the incoming packets, as well as the input source of FPU. The last FIFO is the result buffer, which can also feed the result back to the FPU.
– LSBA & CSBA structures. They are implemented as Content Addressable Memory (CAM) modules. Both are identified by the "*Com*" field of the *reduction packet*, as mentioned above. The CAM width is the sum of the length of LSBA and "*Com*" field, which is no more than (Log$_2$ (NxN) +10) bits. The depth of LSBA-CAM is the maximum number of communicators that it can support at the same time. The depth of CSBA-CAM represents the max number of simultaneous reduction tasks it can support.

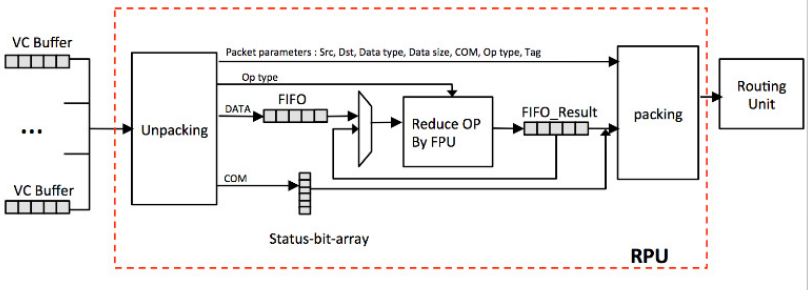


Fig. 4. Block Diagram of RPU

After the RPU has completed all computations, the result packet will be injected into the router-pipeline again; the forwarding direction is dependent on the Y-X deterministic routing algorithm.

- sRPU

As mentioned before, Class_2 is integrated with a simplified RPU that has a smaller status-bit-arrays, which contains only LSBA and no CSBA. Each LSBA is at a fixed length of 5-bit.

3.4 A Processing Example

- Reduction

Take the ADD operation for example. As the first *reduction packet* has been buffered at one of the five input ports of a router, it will be sent to the RPU. The data will be stored in one of the FIFOs of RPU and the result FIFO will be set as the "safe" value (for instance "0" for addition). Then the FPU can execute "ADD" for the incoming and the result data. At the same time, its virtual channel will be released; the original packet will not be forwarded.

When any following packet has arrived, extracted data will be also stored into FIFOs and operated by the FPU to finish the ADD operation with the buffered one; the sum will be stored in the result FIFO to replace the older. Because of the double-buffering mechanism, the data-transmitting and computation can be parallelized.

This procedure will repeat until the LSBA&CSBA have identified that all *reduction packets* have been received, as described in the above subsections. At last, the newly generated data will be packed as a *reduction packet* and sent out.

- *All_reduce*

All_reduce is regarded as a reduction followed by a result-broadcast from the root; the latter is an inverse process of reduction. On reception of such a packet, it will be sent to RPU/sRPU. The control logic will check the LSBA to get the forwarding destinations, which are its neighbors and / or the local core that participated in the reduction of this communicator. For each destination, the RPU/sRPU generates one corresponding packet to send out. This process runs iteratively, until the broadcast finishes.

3.5 Layout of Class_1 and Class_2 Routers

The FPU in RPU is responsible for doing the corresponding computation, which is the main component to consume power and occupy a comparatively large area. As a result, the usage of Class_1 router should be very prudent. On the other hand, fewer Class_1 routers are not enough to achieve high performance. Consequently, the layout of Class_1 and Class_2 routers in a NoC is crucial to both performance and power consumption.

4 Evaluation

4.1 Methodology

To evaluate the effectiveness of the design we proposed, we have implemented a cycle-accurate CMP simulator using the Xtensa Xplorer toolkit [23]. Moreover, the toolkit contains an energy estimator. Thus, for a given design we can get the running cycles / frequency, power consumption and chip area under some CMOS process. Our simulator includes more than one Xtensa LX4 core connected by a NoC module. The Xtensa LX4 can achieve 1.4 GHz on 45nm GS process technology. And the NoC models a detailed pipeline structure for the enhanced routers.

As for NoC, Table 1 lists the network configurations applied to all our experiments. In the experiments, the network scale varying from a 4x4 mesh to a 16x16 mesh in order to measure the scalability.

Table 1. Network Configurations

Network Configuration	Topology	2D mesh
	Routing algorithm	Deterministic Y-X routing
	Channel width/flit size	128 bits (16 Byte)
	Maximum Packet size	5 flits
Enhanced Router	Number of ports	5 ports
	VCs per port	2 VCs
	Buffers per VC	5 flits
	FIFO_A/FIFO_B/ FIFO_Result	5 flits
	Length of LSBA/CSBA	Log_2 (N x N + 10)
	Number of LSBA	5
	Number of CSBA	10
	FPU latency (Multiplication)	6 cycles

Time taken by performing a complete reduction/All_reduce based on our customized NoC is referred as the hardware latency. Also, 3 typical layout strategies are measured in the experiment, shown in Figure 5. The layout of Strategy 1 can be considered as a naive implementation of reduction without any optimization. In this case, only 1 router is the Class_1 router, which is nearest to the center and also designated as the rank root node. While of Strategy 2, there are only one row (1/n of all nodes) that lies in the middle of the network is equipped with Class_1 routers. Strategy 3 is the example shown in Section 3.5, and the number of Class_1 routers is 2/n of all. Because we focus on the small data-set and the enhanced routers can occupy in-transmission reduction packets, the one-pass communication protocol is employed: any core involved in the reduction sends out the *reduction packet* directly without handshake. To be fair, the same strategy is also applied to the software method.

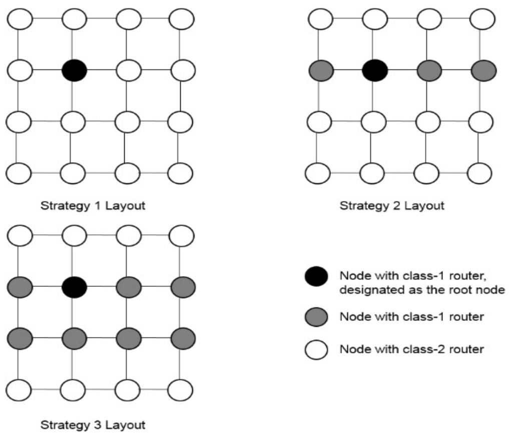


Fig. 5. Three layout strategies

The corresponding time based on the software Binomial Tree Reduce method is referred as the *software latency*. It is estimated on the Log_2P model, while all involved software overheads, such as the packet start-up time, per data transmission time, computation time, etc., are gotten from the real CMP simulation. Moreover, we assume that there is no network contention; thus it can be considered as the greatly optimized reduction implemented by software.

All_reduce is considered as a regular reduce process and then broadcast the result to all nodes from root. The *software latency* of *All_reduce* is estimated base on the Log_2P model as well.

We perform several experiments with different size of payloads of *reduction packet*, as well as network scales. In all cases, the packet size has an upper limit of 64 bytes, which is the size of a L1 cache line. Also, both hardware design of Strategy 1,2 and 3 and software method are evaluated in all experiments.

4.2 Results

- Reduction performance

The *hardware latencies* of different strategies and the *software latency* are shown in Figure 6 of different scales of the NoC, as well as the data payloads of *reduction packet*. It can be inferred from this result that all the three strategies have a great performance improvement than software implementation, for at least 2.67 times. With the 16x16 mesh NoC of Strategy 3, the hardware-based approach achieves its peak speedup of more than 11.76 times improvement over the software-only approach. For all cases, the average is about 8 times.

Analysis shows that for one packet, it takes thousands of cycles by software approach to deal with it. In contrast, although the transmission paths of the hardware method are suboptimal, the high efficiency of hardware implementation makes a good remedy of it.

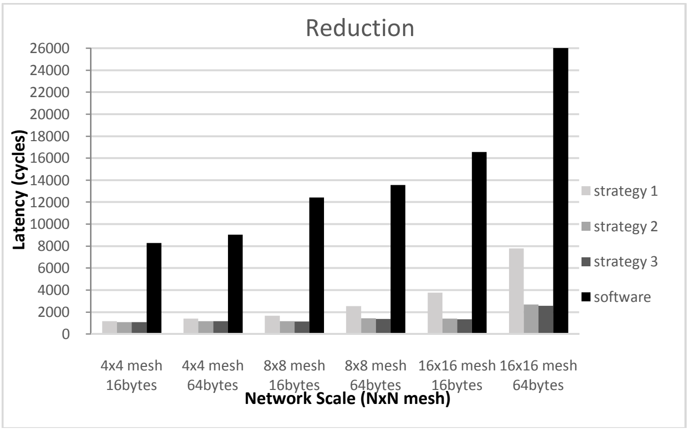


Fig. 6. Reduction latency of hardware and software

As for the hardware optimization, Strategy 3 is slightly better than Strategy 2, and both Strategy 2 and 3 are better than Strategy 1 apparently. Compared with Strategy 1, the least improvement of Strategy 2 and 3 happens at the 4x4 mesh network, which accounts for about 84.4% and 82.9% of the latency of Strategy 1 respectively. This result is reasonable for the small scale network, due to the fact that one Class_1 router has more influence for a small network than for a large. Also, with the increasement of the network scale, the advantage of Strategy 2 and 3 becomes more and more obvious. For the 16x16 mesh topology, Strategy 2 and 3 account for only 24.8% and 22.7% of the latency of Strategy 1 respectively.

Furthermore, the number of transmission hops is greatly decreased for about 50% by Strategy 2 and 3 (compared with Strategy 1), as shown in Figure 7. Transmission of packets contributes most power consumption of NoC, so this is another benefit from our design.

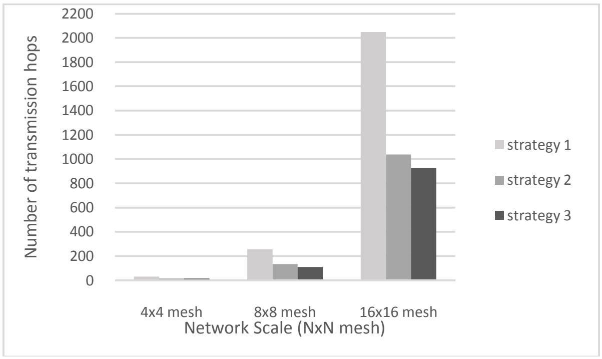


Fig. 7. Number of transmission hops of the three strategies

In consideration of the hardware consumption, the number of Class_1 routers of Strategy 3 is twice as many as Strategy 2's. Therefore, Strategy 2 is the best design strategy, which keeps a balance between the performance and power consumption.

All cores have been involved in the above experiments. In addition, if a reduction task includes fewer nodes and does not contain any Class_1, just as the situation described in Section 3.5, the experiment result shows that it only costs a little more latencies (about 1%~2%) than the case including some Class_1 routers, such as No.18~21 & No.26~29.

- *All_reduce* performance

The speedup of *All_reduce* is shown in Figure 8, compared with software implementation, it has a speedup of 3.35~10.2 times, which is almost the same as reduction. For hardware strategies, the same conclusion with the reduction does still hold.

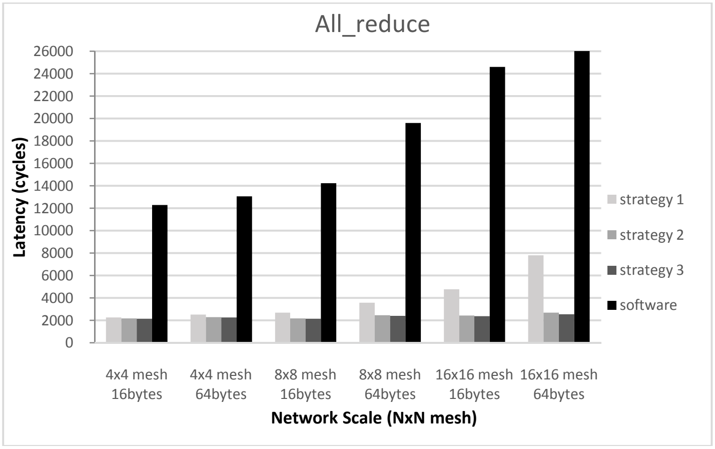


Fig. 8. *All_reduce* result

- Implementation overhead

To obtain the implementation overhead, enhanced routers have been described in HDL and synthesized in 90 nm GS process under typical operating conditions. The area and power have also been measured at the maximum supported frequency. Results show that one Class_1 router is about 90.8% larger than the common router, and the simplified FPU occupies 38% of the total area of a Class_1 router. For power consumption, such a router costs 1.84 times of the common router in one complete reduction. For the Class_2 router, the extra consumption is limited: only 1.4% area overhead is observed.

In the best layout (Strategy 2) mentioned above, only 1/N of routers belong to Class_1 and others are Class_2, which greatly decreases the hardware overhead. More specifically, from the perspective of resource consumption of routers (not including the links and cores), our design has increased 24%, 12.6% and 7% of the chip area for the 4x4, 8x8 and 16x16 mesh-NoCs respectively.

5 Conclusion

The most important contribution of this paper is the two kinds of enhanced routers for message reduction, which are integrated with the Reduce Processing Unit (RPU) that is able to complete reduction and to learn transmission paths adaptively, or a simplified Reduce Processing Unit (sRPU) that is only capable of learning transmission paths. Three implementation of layout strategies of the customized network-on-chip is presented and evaluated to prove the effectiveness. Compared to the most efficient software-based implementation of reduce/all-reduce operation, we achieve improve the performance of 2.67~11.76 times (up to 11.76 for reduction and 10.2 for *All_reduce*). In addition, the extra chip area and power dissipation has been greatly reduced by the optimal strategy.

Acknowledgment. The work is supported by the High Tech. R&D Program of China under Grant No. 2013AA01A215.

References

1. Timothy, M.: The Future of Many Core Computing, `http://i2pc.cs.illinois.edu/presentations/2010_05_06_Mattson_Slides.pdf`
2. Rakesh, K., Timothy, G.M., Gilles, P., Rob, V.D.W.: The Case for Message Passing on Many-Core Chips. Multiprocessor System-on-Chip, pp. 115–123 (2011)
3. Jie, M., Daniel, R., Ayse, K.C.: 3D Systems with On-Chip DRAM for Enabling Low-Power High-Performance Computing. In: Proceedings of Fifteenth HPEC Workshop, Massachusetts, USA (September 2011)
4. Timothy, G.M., Rob, F.V.D.W., Michael, R., Thomas, L., Paul, B., Werner, H., Patrick, K., Jason, H., Sriram, V., Nitin, B., Greg, R., Saurabh, D.: The 48-core SCC processor: the programmer's view. In: Proceedings of 2010 International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA (2010)
5. MULTICORE COMMUNICATIONS API WORKING GROUP, `http://www.multicore-association.org/workgroup/mcapi.php`
6. Dong, Y., Chen, J., Yang, X., Yang, C., Peng, L.: Low power optimization for MPI collective operations. In: The 9th International Conference for Young Computer Scientists, ICYCS 2008, IEEE (2008)
7. Rabenseifner, R.: Automatic MPI counter profiling of all users: First results on a CRAY t3e 900-512. In: Message Passing Interface Developer's and User's Conference (1999)
8. Rabenseifner, R.: Optimization of collective reduction operations. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3036, pp. 1–9. Springer, Heidelberg (2004)
9. Open MPI Development Team, Open MPI: open source high-performance computing, `http://www.open-mpi.org/`
10. Thakur, R., Rabenseifner, R., Gropp, W.: Optimization of collective communication operations in MPICH. High Performance Computing Applications 19(1), 49–66 (2005)
11. Rabenseifner, R.: Optimization of collective reduction operations. In: Proceedings of Int'l Conference on Computational Science (ICCS), Krakow, Poland (2004)
12. Nicolas, F., Marc, H., Eric, L., Bernard, T.: MPI for the Clint Gb/s Interconnect. In: Proceedings of the 10th European PVM/MPI User's Group Meeting, pp. 395–403 (2003)

13. Maximize Platform MPI Performance with Voltaire® Fabric Collective Accelerator™ (FCA™) and HP, `http://www.mellanox.com/related-docs/voltaire_acceleration_software/FCA-Voltaire-Platform-HP-WEB111110.pdf`
14. Underwood, K.D., Ligon, W.B., Sass, R.R.: Analysis of a prototype intelligent network interface. Concurrency and Computation: Practice and Experience 15(7-8), 751–777 (2003)
15. Almási, G.S., et al.: Implementing MPI on the blueGene/L supercomputer. In: Danelutto, M., Vanneschi, M., Laforenza, D. (eds.) Euro-Par 2004. LNCS, vol. 3149, pp. 833–845. Springer, Heidelberg (2004)
16. Gao, S., Schmidt, A.G., Sass, R.: Impact of reconfigurable hardware on accelerating mpi_reduce. In: 2010 International Conference on Field-Programmable Technology (FPT), pp. 29–36 (2010)
17. Libo, H., Zhiying, W., Nong, X.: Accelerating NoC-based MPI Primitives via Communication Architecture Customization. In: Proceedings of IEEE 23rd International Conference on Application-Specific Systems, Architectures and Processors, Delft, July 2012, pp. 141–148. IEEE (2012)
18. David, W., Patrick, G., Henry, H., Liewei, B., Bruce, E., Carl, R., Matthew, M., Chyi-Chang, M., John, F.B., John III, F.B., Anant, A.: On-chip Interconnection Architecture of the Tile Processor. IEEE Computer Society (September-October 2007)
19. Velamati, M.K., Kumar, A., Jayam, N., Senthilkumar, G., Baruah, P.K., Sharma, R., Kapoor, S., Srinivasan, A.: Optimization of collective communication in intra-cell MPI. In: Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2007. LNCS, vol. 4873, pp. 488–499. Springer, Heidelberg (2007)
20. Ali, Q., Midkiff, S.P., Pai, V.S.: Efficient high performance collective communication for the cell blade. In: Proceedings of the 23rd International Conference on Supercomputing, pp. 193–203. ACM (2009)
21. Kohler, A., Radetzki, M., Gschwandtner, P., Fahringer, T.: Low-latency collectives for the intel scc. In: 2012 IEEE International Conference on Cluster Computing (CLUSTER), pp. 346–354. IEEE (2012)
22. Peng, Y., Saldaña, M., Chow, P.: Hardware support for broadcast and reduce in mpsoc. In: 2011 International Conference on Field Programmable Logic and Applications (FPL), pp. 144–150. IEEE (2011)
23. Gonzalez, R.E.: Xtensa: A configurable and extensible processor. IEEE Micro 20(2), 60–70 (2000)

Athena: A Fault-Tolerant, Efficient and Applicable Routing Mechanism for Data Centers

Lijun Lyu, Junjie Xie, Yuhui Deng*, and Yongtao Zhou

Department of Computer Science, Jinan University, Guangzhou, P.R. China
{lvlijun1992,xiejunjiejnu}@gmail.com,
tyhdeng@jnu.edu.cn, y.t.zhou@foxmail.com

Abstract. The overall performance of data center depends on the physical topology and the corresponding routing mechanism. Many novel network structures have been proposed in recent years to remedy the shortcomings of traditional tree-based structure. Especially some hybrid recursively defined structures with acceptable costs can perform well. These structures mainly adopt the conventional routing mechanism which maintains large and complex link states. However, this routing mechanism still can not work out the cost-optimal path to meet the requirement of short latency and low extra traffic consumption. Hence, this paper presents Athena Routing Mechanism (ARM) based on Dynamic Programming with path probing scheme to further promote the performance of those structures. ARM is fault-tolerant since it makes full use of redundant links. It is also able to work out the shortest paths, which shortens the communication delay and releases intermediate servers from forwarding loads as well as extra CPU and bandwidth resources. Results from theoretical analysis, simulations and experiments firmly support the conclusion that ARM is a fault-tolerant and efficient routing mechanism which is able to be generalized to many other hybrid structures.

Keywords: Data Center, Interconnection Network, Routing Mechanism, Fault Tolerance.

1 Introduction

The continuous rise and great foreground of cloud computing creates incredible amounts of data, promoting the status of data center to a new height in recent years. With considerable investments of funds and labor, data center develops rapidly as an increasing scale. However, in a large-scale data center, failures from software, hardware or even outage become quite ubiquitous [1] [2]. Moreover, in cloud computing, data center provides increasing online application services and network capacity has a significant impact on user experience. Therefore, a desirable data center networking should meet these three requirements. First, the network infrastructure must be scalable and enable incremental expansion.

* Corresponding author.

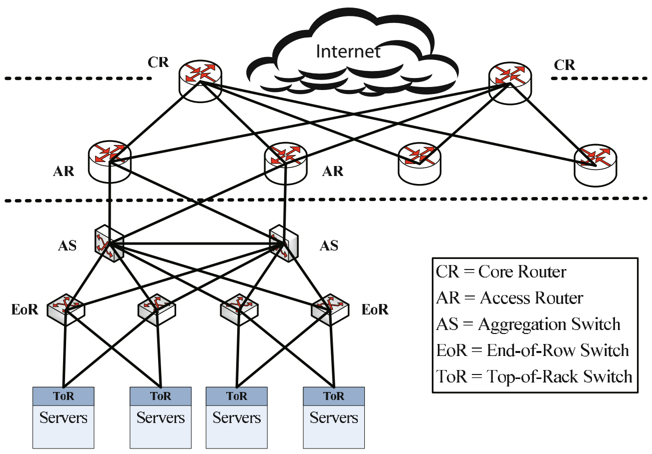


Fig. 1. Traditional Tree-based Structure with 3 levels

Second, the data center networking must be fault tolerant against various types of failures. Last, in purpose of better supporting bandwidth-hungry services, the high network capacity and short latency are also requested.

The current data centers are universally tree-based. Servers in the same rack are connected with a Top-of-Rack (ToR) switch. Then ToRs are connected to End-of-Row (EoR) switches which are finally linked with core switches or routers. Internet services are provided by a tree hierarchy of core switches or core routers. Practices have proven the inherent vulnerabilities of this structure, especially the bandwidth bottleneck and single point of failure [3]. Besides, tree-based structures commonly adopt the classical routing mechanisms, which are proven valid in traditional Internet. However, these mechanisms still can not meet the requirements of bandwidth and latency for data center since they do not make full use of the linking characteristics of data center. Moreover, the expensive top-level devices will cause sharp rise of costs and restrain flexible expansion of physical structure. Thus it is significant to build a new network structure that can address these issues fundamentally in both the physical network architecture and the protocol design. Hitherto, many researches proposed plentiful feasible structure schemes, such as Fat-Tree [3], which is an improved tree-based structure. And there are some other hybrid and recursively defined structures like *DCell* [4], *FiConn* [5], *BCube* [6] and Totoro [7]. These structures put linking and routing intelligence on servers instead of switches. Thus costly core switches and routers can be replaced by cheap commodity switches. Their high-level structures can be constructed by several low-level structures. This feature of recursive definition makes the physical structure scales exponentially, which meets the first requirement mentioned above.

As to fault tolerance and network capacity, both of which are fulfilled by reasonable architecture and routing algorithm. In the aforementioned hybrid structures, multi-ports servers and COTS (commodity off-the-shelf) mini-switches are utilized to build multiple redundant links. In order to balance the reliability and cost, tradeoffs are inevitable. For example, Bcube maintains favorable connectivity between any two nodes, but it costs a huge number of wires and switches. On the contrary, FiConn has less redundant links and switches, and thus it costs less. Their respective routing algorithms mainly take charge of bypassing failed links and balancing network load. Therefore, the routing algorithm plays a vital role in achieving high fault tolerant and network capacity.

One of the conventional routing policy is breaking the whole network into equivalent broadcast domains. The aforementioned DCell and Totoro both adopt this policy. A broadcast domain is a substructure. Servers in the same domain preserve link states of each other. And the link states between two different domains are maintained by outer-servers, which connect to a domain with outgoing links. All link states information needs to be updated at short intervals in case of any change or failure. Since the failures in data centers occur commonly, frequent data exchanges in network structure may be conducted, resulting to unnecessary consumption of bandwidth. During the routing process, every node executes routing algorithm to find a next hop to the final destination. This might cause a relatively high CPU usage for all servers comprising the path. For instance, the CPU utilizations are all over 40% for sender, receiver and forwarder in DCell [4]. Therefore, this broadcast policy is not very satisfactory. Actually, we can figure out the completed paths by only one time computation and release intermediate servers from repetitive computation.

From the above, we offer another option of routing mechanism to optimize network performance. We name it *Athena Routing Mechanism* (ARM), which can be generalized to most other data center structures mentioned before. The basic principle of ARM is transferring the responsibility of path calculation on source server, the intermediate nodes only take charge of transmitting data packets. The validation and capacity of a path are confirmed by path probing scheme, thus broadcast can be omitted for saving bandwidth. Besides, our routing algorithm is based on *Dynamic Programming* (DP), hence the shortest paths can be worked out primarily in path calculation process. The selection of paths and load balancing are achieved by path probing procedure. In the following parts, we will present the fundamental theory and implementation of our ARM. We will also prove the high efficiency and superior fault tolerant capability of ARM at the support of extensive simulations. Moreover, our ARM is able to be generalized to most other hybrid network structures mentioned before. In this paper, we choose Totoro as the physical network architecture to test the performance of ARM.

The rest of the paper is organized as follows. Section 2 introduces the related work predecessors have achieved. Section 3 describes the Totoro structure briefly. Then section 4 elaborates on Athena Routing Mechanism. Section 5 and Section 6 use implementations and simulations to evaluate ARM. Lastly, Section 7 concludes the paper.

2 Related Work

The study of interconnection networks have thrived for decades. In recent years, many scholars dedicate themselves to this field in order to achieve a data center interconnection with higher performance. As we know, there are two main determinant factors: a physical structure and a routing algorithm. In general, the routing algorithm is attached to the corresponding structure. In this paper, we will emphasize on the routing algorithms of hybrid structures.

Based on a *Divide-and-Conquer* approach, *DCellRouting* [4] firstly calculates the intermediate links which interconnects two substructures comprising the source and destination servers. Then a "left" sub-path and a "right" sub-path can be worked out recursively to form a completed path. In addition, DCell also adopts a broadcast scheme by dividing the whole network into broadcast domains. Hence the Dijkstra algorithm [8] can replace DCellRouting to find the next hop with shortest length in a broadcast domain. Moreover, there are Local-reroute and Jump-up policy assisting DCellRouting to achieve a high fault tolerant capacity. Namely, if an intermediate server fails to find a next hop by DCellRouting, data packets will be transmitted to a proxy server connects with the intermediate server through an equivalent or higher level of link. Totoro routing mechanism [7] shares the similar routing principle with DCell, so we will not repeat here.

BCubeRouting [6] finds a path from a source to a destination in a BCube structure by correcting one digit at one step to systematically build a series of intermediate servers. Since two neighboring servers which connect to the same level-i switch only differ at the ith digit in the address arrays. And the number of different digits of two address arrays is $k+1$ at most (k is the structural level), thus the longest shortest path length between any server pairs of a $BCube_k$ is $k+1$. Nevertheless, this low-diameter feature is benefited from the corresponding structure with considerable wiring cost, which may not be very suitable to be generalized to other more economic structures.

The *Traffic-Oblivious Routing* (TOR) [5] of FiConn is also recursively defined to make use of the level-based feature. Each intermediate server runs TOR to find the next hop by lowest common level. TOR balances the use of different levels of links and servers. In order to further balance the traffic volume, *Traffic-Aware Routing* (TAR) is proposed. TAR utilizes a greedy approach to hop-by-hop setup of traffic-aware path on each intermediate server. That is, the intermediate servers always select the outgoing link with higher available bandwidth to forward the traffic. In a $FiConn_i$, if the outgoing link found by TOR has lower bandwidth than the other links, then it will be bypassed via randomly selecting a third $FiConn_i$ in $FiConn_{i+1}$ to relay the traffic. In order to avoid the considerable overhead caused by exchanging traffic states among servers, FiConn adopts a probing policy which is analogous to our ARM to establish routing entry for data flow. Even though, the path length of FiConn might be a weakness comparing with ARM.

In a word, except BCubeRouting, all routing algorithms above require intermediate servers involved in the routing process. The repetitive computing will

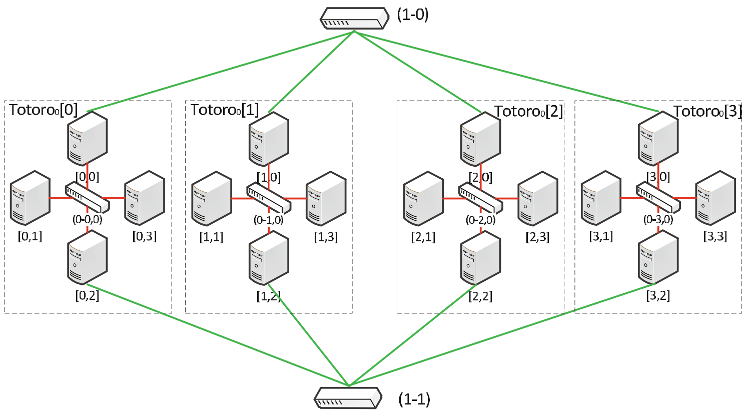


Fig. 2. A $Totoro_1$ structure with $n = 4$

undoubtedly cause unnecessary burden for intermediate servers. Besides, there is still room to promote in path length for those routing algorithms. Therefore, our ARM can be a desirable solution to address the above problems for hybrid and cost-efficient data center networks.

3 Totoro Structure

In order to evaluate the performance of our Athena Routing Mechanism, we simulate it on the physical framework of Totoro and compare with the original *Totoro Fault Tolerant Algorithm* (TFR) and *Shortest Path Algorithm* (SPA, based on Floyd-Warshall [9]). TFR is the original Totoro Fault Tolerant Algorithm, which broadcasts link status in a domain and calculates the routing path hop-by-hop. This section mainly presents the physical structure of Totoro.

Totoro structure consists of a series of commodity servers with dual ports and low-end commodity switches. The basic partition of Totoro is denoted as $Totoro_0$, constructed by n servers connecting to an n-port switch. As mentioned before, Totoro is a hybrid structure with recursive definition. A $Totoro_i (i > 0)$ is constructed from n $Totoro_{i-1}s$. Each round of construction consumes half of the total available ports, and the rest half are remained for expansion. As Fig. 2, a $Totoro_1$ structure with $N = 4$, $n = 4$ is composed of 4 $Totoro_0s$. Each $Totoro_0$ has 4 servers and an intra-switch with 4 ports. 4 $Totoro_0s$ connect through 2 inter-switches. Unlike DCell and FiConn, there are duple direct links between two equivalent substructures, thus the redundant links can be fully used for distributing data flows. Please refer to [7] for details.

A server in Totoro can be indicated in two ways: *Totoro tuple* and *Totoro ID*. Totoro tuple is a $(K+1)$-tuple $[a_K, a_{K-1}, ..., a_i, ..., a_1, a_0]$, which indicates where this server is located. Totoro ID is an unsigned integer, taking a value from $[0, t_K)$ (t_K is the total number of a $Totoro_K$). The mapping between Totoro tuple and Totoro ID is bidirectional.

Intuitively, the physical structure of Totoro is rather symmetric and homogeneous. This feature makes it very suitable for performing distributed file system, such as *Hadoop File System* (HDFS) [10] [11]. Since HDFS requires multiple replicas in different racks and shorter distances between different nodes, there are n servers located in n different racks exactly connecting with each other at each level in Totoro. Thus replicas distributed in different racks maintain the shortest paths between each other. And our routing algorithm endeavors to find out the shortest paths. In combination of our new routing algorithm, both the reliability and efficiency of HDFS will be highly promoted.

4 The Athena Routing Mechanism (ARM)

As we know, the general routing algorithm is based on broadcasting link status in a broadcast domain, and the source and intermediate servers do repetitive computation to find a next hop continually with the link status. This policy causes considerable waste of computation and network bandwidth inevitably. Hence we propose another routing mechanism called Âthena Routing Mechanism (ARM) to address this problem. Since the physical structure is computable, the path to any destination host can be worked out by the source host solely. Intermediate nodes only take charge of forwarding data packets. Before moving to the detailed implementation of ARM, we first focus on presenting the basic algorithm ,*Athena Routing Algorithm* (ARA), which is conducted by the source host.

4.1 Athena Routing Algorithm (ARA)

ARA is based on Dynamic Programming (DP) to obtain simplicity and efficiency. DP breaks a problem into several simpler subproblems. It is applicable to problems which exhibit the properties of overlapping subproblems and optimal substructure. We present how we recursively work out constant number of shortest paths in Algorithm. 1. The function **getLCL** returns the lowest common level u of two nodes (Line. 6.). Then the function **getTopLinks** (Line. 11.) figures out all level-u links starting from the whole level-u substructure which the source is located. Afterwards, for each independent top link, we can recursively find a set of completed paths from the source server to the destination server. What noteworthy is the total independent paths might share same nodes, even though we choose absolutely different links in every round of recursion. This is because low-level pahts may share the same high-level path. So when one path fails, we can not assure all the other paths are unaffected. This is a tradeoff to facilitate the routing algorithm and we can alleviate this problem by detecting multiple paths simultaneously.

Take Totoro structure as an exemplification, in Fig. 2, server [0,1] need communicate with server [1,3]. To calculate all completed paths by ARA is as follows: Firstly, the lowest common level u of these two servers is 1. Then we find two level-1 links $[0,0] \rightarrow [1,0]$ and $[0,2] \rightarrow [1,2]$. Afterwards, take the top-level

Algorithm 1. Athena Routing Algorithm

```
1: //param $count$: denotes the upper-bound number of paths to return
2: function AROUTE($src$, $dst$, $count$)
3:     if $src == dst$ then
4:         return $NULL$
5:     end if
6:     $u$ = getLCL ($src$, $dst$) //lowest common level
7:     if $u == 0$ then //in the same basic structure
8:         return P ($src$, $dst$)
9:     end if
10:    //level-u links between two substructures
11:    Set $topLinkSet$ = getTopLinks ($src$, $dst$, $u$)
12:    Set $result$
13:    for each link $L$ in $topLinkSet$ do
14:        $leftPathSet$ = ARoute ($src$, $L.leftNode$, $count$)
15:        $rightPathSet$ = ARoute ($L.rightNode$, $dst$, $count$)
16:        $result$.add ($leftPathSet + L + rightPathSet$)
17:    end for
18:    SortByLength (result)
19:    $result$ = $result$.sublist (0,$count$)
20:    return $result$
21: end function
```

links as root path respectively to find out left paths and right paths recursively. Therefore, we can get two completed paths: $[0,1] \rightarrow [0,0] \rightarrow [1,0] \rightarrow [1,3]$, $[0,1] \rightarrow [0,2] \rightarrow [1,2] \rightarrow [1,3]$. Since DCell and FiConn have the similarly level-based structures as Totoro, ARA can be easily applied to them. The only difference should be that the **getTopLinks** function is related to the detailed structural characteristics. For simplicity, we do not present the details here.

4.2 Path Probing of ARM

The basic idea of Athena Routing Mechanism is figuring out all completed paths by ARA before sending probing packet (Actually we need not find all paths in practice. A threshold value can be utilized to limit the number of paths). If the source host *src* maintains a path cache to the destination host *dst*, this step can be omitted. Then a set of probing packets will be dispatched to detect the capacities of selected paths. The intermediate servers record the link capacities in the probing packets and forward them to the destinations by the pre-calculated paths. After the probing packets arrive, the destination servers reply these probing requests by sending the probing packets back to the source hosts according to the original paths. Among all the valid paths, we tend to choose the one with higher bandwidth and/or shorter length, so that we can get sufficient resources every time and utilize links evenly.

4.3 Properties of ARM

In general, our ARM performs well in Totoro. Primarily, our routing algorithm works out all paths directly connecting two substructures, so no circuitous path is involved. With the feature of DP, we can ensure the shortest path length. Besides, our ARA takes consideration of all top-level switches connecting the substructures which *src* and *dst* are located respectively. Thus we can get sufficient feasible paths by ARA to achieve a desirable fault tolerant capacity. Moreover, the path probing policy of ARM is able to save a great number of bandwidth for Totoro, since the size of probing packet is far less than broadcast data packet among all servers. The intermediate servers are also released from heavy routing computation loads because they only carry on forwarding data packets.

Theorem 1. *n represents the number of servers in Totoro. k is the level of Totoro structure. T_k is denoted as the total complexity. The time complexity of ARA is $O(T_k) \leq O(N^{\frac{k}{2}})$.*

Proof. The number of top-level links ARA worked out firstly is equal to the number of top-level switches, that is $(n/2)^k$. And then two recursion process are conducted. Therefore, we can get $T_k = 2 \times T_{k-1} \times (n/2)^k$. According to mathematical induction, we can finally figure out the equation

$$T_k = \frac{N^{\frac{k}{2}}}{2^{\frac{k(k-1)}{2}}} \tag{1}$$

So omit the denominator (since it is lager than 1), theorem 1 is rigidly proved. Note that k is a small integer, a low-level Totoro (e.g., $n = 32, k = 3$) still can support more than one million servers ($32^{3+1} = 1048576$). Thus the time complexity of ARA is relatively low when k is small. □

Besides the conventional tree-based structures, subsequent scholars have proposed many structured network interconnections. As we mentioned before, DCell, FiConn and BCube are all with this feature. That is, the pathes from a source host to a destination host can be totally figured out by the source according to the physical properties of the architecture. Moreover, with ARA, we can obtain the shortest path set with lower time complexity than that of SPA, hence all above structures will get a huge promotion of average path length. Besides, especially for DCell, which adopts broadcasting link status among servers, our path probing mechanism will efficiently save network bandwidth as well as relieve computation load for intermediate servers. All in all, our Athena Routing Mechanism is able to be applied to many other structured network interconnections. Comparing with there original routing algorithms, our proposal can also achieve desirable efficiency and fault tolerant capability.

5 Athena Protocol Implementation

5.1 ARM Address

Since most applications are based on TCP/IP, we then design ARM protocol as a 2.5-layer protocol. In adaptation of Totoro structure, we represent a specific

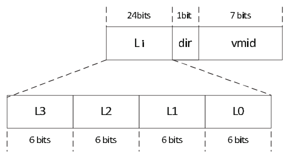


Fig. 3. Address Format

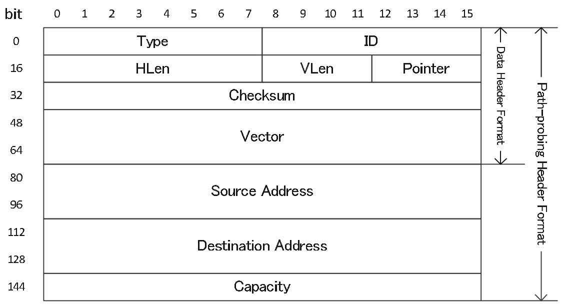


Fig. 4. Header Format

server by a 32-bits tuple named ARM Address. Since it has the same length with IP address, we then utilize this tuple in place of IP address in the IP header, i.e., we set the IP address to the same value of ARM address. Thus we can use the source and destination address from IP header directly, rather than add two additional fields.

As Fig. 3, there are three fields in ARM Address: L_i, *dir* and *vmid*. L_i denotes the server position in the network. In this paper, we suppose i is no more than three so that L_i consists of tuples from L_0 to L_3 with 6-bits length each. Actually, i indicates the level of Totoro structure. Note that a 4-level Totoro structure ($k = 3, n = 48$) can support as many as five millions servers. And we can simply complete the high-order position or adjusting length of each L_i field to apply to a smaller structure with less servers. The *dir* takes up one bit to indicate this port connects to an intra-switch or inter-switch. *vmid* means the index of virtual machine in a physical server. It occupies seven bits so that we can support 127 virtual machines at most ($vmid = 0$, represents the physical server itself). We set this field only for adapting the trend of cloud computing, and it will not be used in the routing computing since only physical addresses are involved in our ARM and when data packet arrives at the target physical server, it will be transferred to the specific virtual machine by operation system.

5.2 Packet Format

There are two types of packet: path-probing packet and data packet. Before dispatching a data packet, a set of path-probing packets will be delivered first to confirm the capacities of selected paths. It involves the source address and destination address, as well as the capacity of one path and so forth. The difference between path-probing packet and data packet is the former involves the fields of source and destination address as well as capacity. Fig. 4 shows the format of packet header of these two types of packets.

5.3 ARM Protocol

As the fact that two adjacent servers in a path only differ at one "bit" (i.e., one item in Totoro tuple), we adopt the path transformation vector to preserve path information. A vector is included in a data packet, and each item of this vector represents "one-bit" change. An intermediate server determines the next hop according to the value of item which current pointer is pointing at. This approach of preserving the path information in a vector has been adopted in former research [12]. If a server receives a packet from the upper layer, it first checks whether the destination address is a loop address or not. If so, then it returns the packet to the upper layer. Otherwise, the server checks if it maintains a cache of path information to the destination. If not, it then employs ARA to figure out a set of paths and update path cache. Then the probing packet of request is constructed, in which the path transformation vector is also initialized. Afterwards, the probing packet is dispatched, intermediate servers forward it to the next hop according to the vector. When it arrives at the destination host, the host will send back a reply message along the previous path. Then a data packets will be dispatched to the destination along the selected path.

6 Experiments and Result

In Fig. 5 to Fig. 8, we evaluate the path failure ratio of Totoro using ARM under four types of failure,including link failure, server failure, switch failure and rack failure. We run ARA, TFR and SPA on a $Totoro_2$ $(n = 16, k = 2, t_k = 4096)$ under those four types of failures. Meanwhile, the results are compared with TFR and SPA. A rack consists of a $Totoro_0$. Failures are generated randomly ranging from the ratio of 2% to 20%. Servers deliver packet to other nodes 20 times, Which means every final result is the average of 20 running results.

Before presenting our experiment results, we will introduce TFR and SPA briefly. By TFR, servers randomly choose a nearest level-u (u denotes the lowest common level with destination) link to the next hop, proceeding this process recursively until finding the destination. If failure occurs, another level-u or even higher links will be adopted. In addition, Totoro breaks the whole network into broadcast domains (*TBD*). In a TBD, link status are exchanged by broadcast among all servers. Thus the Dijkstra algorithm can be performed in a TBD to shorten path length. SPA is based on Floyd-Warshall algorithm, which requires global link states information to find out the shortest path. Consequently, SPA is globally optimal whereas the time complexity of it is as high as $O(N^3)$(N denotes the total number of servers in a Totoro structure).

As Fig. 5 and Fig. 6 indicate, the path failure ratios of the all three algorithms are equivalent under server and rack failure respectively. That is, the fault tolerant capacity of ARM is almost optimal. It also proves that ARM makes full use of redundant links and switches between two substructures. Rack failure means all servers in a rack are invalid, so it is analogous to server failure. In Fig. 7, under switch failure scenario, ARM performs much better than the original TFR. From Fig. 8, we can see when a high link failure occurs, ARM achieves slightly better fault tolerant capacity than TFR. But compared with SPA, ARM is still a bit inferior. This makes sense since all top-level links calculated in Totoro are direct links between two substructures. On the contrary, SPA will traverse all feasible

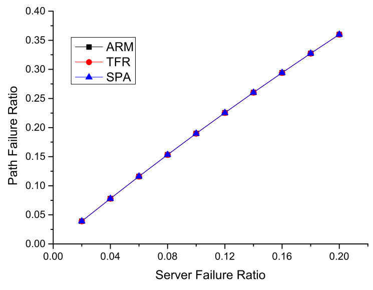


Fig. 5. Path vs. Server failure ratio.

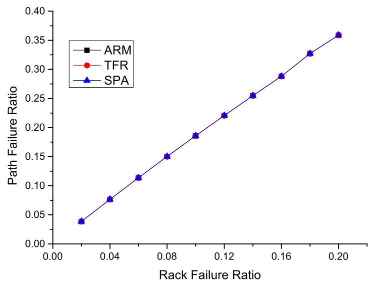


Fig. 6. Path vs. Rack failure ratio.

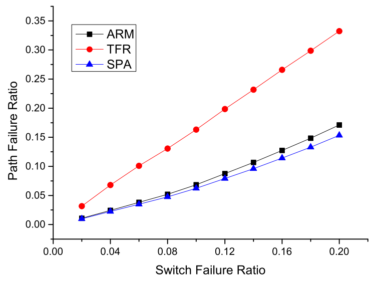


Fig. 7. Path vs. Switch failure ratio.

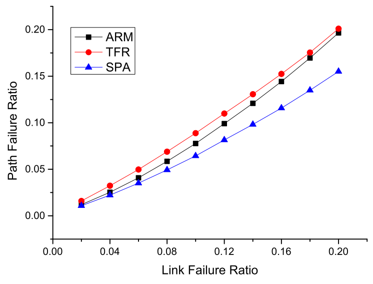


Fig. 8. Path vs. Link failure ratio.

links in the whole structure until finding a valid path. Hence this is a trade-off that ARM makes to facilitate algorithmic complexity and save computation resources.

The efficiency of routing algorithm can be directly evaluated by path length. A short average path length contributes to a short latency. Table. 1 lists the values of average path length calculated by ARM, TFR and SPA respectively for a $Totoro_2$ $(n = 16, k = 2, N = 4096)$. We take SPA as the benchmark of routing performance because SPA is globally optimal. Comparing the results of ARM and SPA, it is easy to draw a conclusion that the differences are negligible. In some cases, such as server failure and rack failure, the average path lengths of both are equivalent. Whereas in link and switch failure, our ARM achieves shorter path lengths than SPA does, this is because the path failure ratio of ARM is a bit higher than that of SPA, thus our total path length is shorter. Besides, ARM is much simpler than SPA, for the latter's computation complexity is as high as $O(N^3)$, and it requires frequent exchanges of link status, which may cause heavy loads for data center. In conclusion, the failure experiments prove the great fault tolerant capacity as well as high efficiency of our Athena Routing Mechanism.

Table 1. Average Path Lengths in $T_{16,2}$

Failure Ratio		0.04	0.08	0.12	0.16	0.20
Server Failure	TFR	8.44	8.48	8.52	8.57	8.63
	ARM	7.34	7.41	7.48	7.56	7.64
	SPA	7.34	7.41	7.48	7.56	7.64
Link Failure	TFR	8.98	9.68	10.64	11.97	13.69
	ARM	7.47	7.68	7.90	8.16	8.47
	SPA	7.47	7.69	7.94	8.22	8.54
Switch Failure	TFR	8.43	8.46	8.50	8.53	8.58
	ARA	7.39	7.53	7.67	7.82	7.97
	SPA	7.40	7.55	7.70	7.86	8.03
Rack Failure	TFR	8.54	8.69	8.85	9.02	9.23
	ARM	7.33	7.40	7.48	7.57	7.66
	SPA	7.33	7.40	7.48	7.55	7.65

In Fig. 9, we also append the CPU usage of committing routing algorithm in source server to evaluate the computing efficiency. The experiment environment is under a Lenovo T350 G7 server with quad-core processors and 8GB memory. We simulate a $Totoro_2$ ($n = 16, k = 2, N = 4096$), and run ARM on the experimental server to work out 10 completed paths with any node in the same or neighbor 3 $Totoro_1$ as the destination. Because of data locality, a server usually communicates with servers which are located in the same row or adjacent several rows. We set the initial nodes amount as 10 and increase by 10 per second until reaching the threshold value 500, and the total computing time is 3 minutes.

As Fig. 9 indicates, the CPU usage continuously increases until achieving the peak value of 28% at around the 20_{th} second. Afterwards, it dramatically drops to 0% and remains to the end. Thus we can get a conclusion that our ARM has great performance with rather low CPU usage. Besides, the dash line represents the number of nodes which the experimental server calculates per second. From the graph, we can see the server figures out 10 paths of 500 nodes per second in the cost of 0% CPU usage, this is benefited from path cache constructed in the source server. Suppose a server maintains a cache of 10 completed paths to all of the rest nodes in the network. The maximal length of a path consists of 5 hops, and the vector of a path take up $6bits \times 5hops = 30bits$ memory, that is $4B$ approximately. Then we can get the largest size of cache to preserve all path information on each host is $4B \times 10 \times 4096 \approx 164KB$ at most. This also further proves our ARM is resource saving.

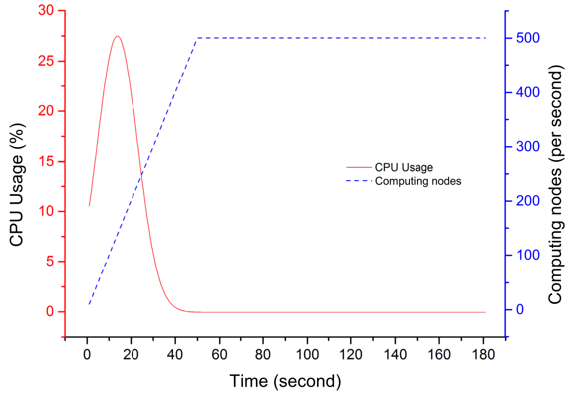


Fig. 9. Resource Usage

7 Conclusion

In this paper, we have presented a routing mechanism with favorable fault tolerant capacity and high efficiency. For hybrid and server-centric data center structures using inexpensive commodity switches, there are still room to improve in there respective routing algorithm. And this motivates us to propose a universally applicable routing mechanism named ARM comprising of a routing algorithm called ARA and a path probing scheme. On basis of Dynamic Programming (DP), ARA is able to find out the shortest paths with far lower time complexity than SPA. ARM adopts source routing and path probing scheme, which means the source server performs computing a set of completed paths and dispatchs probing packets to detect the connectivity and capacities of paths. ARM is implemented by a 2.5-layer protocol and a 32-bits ARM address. In comparison with other conventional routing policies, our ARM simplifies the functionalities of intermediate servers as well as eliminates the extra bandwidth consumption caused by broadcasting link states among servers. Besides, our failure experiments on Totoro structure prove the satisfactory fault tolerant capacity of ARM comparing with TFR and SPA under different types of failures. We also demonstrate the relatively low CPU usage ratio of source server during the path computing process. Therefore, our ARM is a reliable and efficient mechanism which can be generalized to most hybrid structures to promote the overall performance of data center network. In the future work, we will focus on the implementation of ARM in DCell, FiConn and other structures to further verify the performance of our ARM.

Acknowledgment. We would like to thank the anonymous reviewers for helping us refine this paper. Their constructive comments and suggestions are very helpful. This work is supported by the National Natural Science Foundation (NSF) of China under grant (No.61272073, No. 61073064), the key program of Natural Science Foundation of Guangdong Province (no. S2013020012865), the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry), the Educational Commission of Guangdong Province (No. 2012KJCX0013). The corresponding author of this paper is Yuhui Deng.

References

1. Gantz, J.F., Chute, C.: The diverse and exploding digital universe: An updated forecast of worldwide information growth through 2011. In: IDC (2008)
2. cnbeta: Ten cloud crashes in 2013 (2014), `http://www.cnbeta.com/articles/266790.htm`
3. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. ACM SIGCOMM Computer Communication Review 38, 63–74 (2008)
4. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. ACM SIGCOMM Computer Communication Review 38, 75–86 (2008)
5. Li, D., Guo, C., Wu, H., Tan, K., Zhang, Y., Lu, S.: Ficonn: Using backup port for server interconnection in data centers. In: INFOCOM 2009, pp. 2276–2285. IEEE (2009)
6. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: Bcube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Computer Communication Review 39(4), 63–74 (2009)
7. Xie, J., Deng, Y., Zhou, K.: Totoro: A scalable and fault-tolerant data center network by using backup port. In: Hsu, C.-H., Li, X., Shi, X., Zheng, R. (eds.) NPC 2013. LNCS, vol. 8147, pp. 94–105. Springer, Heidelberg (2013)
8. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik 1(1), 269–271 (1959)
9. Floyd, R.W.: Algorithm 97: shortest path. Communications of the ACM 5(6), 345 (1962)
10. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
11. Borthakur, D.: The hadoop distributed file system: Architecture and design. Hadoop Project Website 11, 21 (2007)
12. Riley, G.F., Ammar, M.H., Zegura, E.W.: Efficient routing using nix-vectors. In: 2001 IEEE Workshop on High Performance Switching and Routing, pp. 390–395 (2001)
13. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications 1(1), 7–18 (2010)
14. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. ACM SIGCOMM Computer Communication Review 39(1), 68–73 (2008)

Performance-Aware Data Placement in Hybrid Parallel File Systems

Shuibing He[1,2], Xian-He Sun[2], Bo Feng[2], and Kun Feng[2]

[1] School of Computer, Wuhan University, Wuhan, Hubei, China
[2] Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois, USA
{she11,sun,bfeng5,kfeng1}@iit.edu

Abstract. Hybrid parallel file systems (PFS), which consist of both HDD and SSD servers, provide a promising solution for data-intensive applications. In this study, we propose a performance-aware data placement (PADP) strategy to enable efficient data layout in hybrid PFSs. The basic idea of PADP is to dispatch data on different file servers with adaptive varied-size file stripes based on the server storage performance. By using an effective data access cost model and a linear programming optimization method, the appropriate stripe sizes for each file server are determined effectively. We have implemented PADP within OrangeFS, a widely used parallel file system in HPC domain. Experimental results of representative benchmark show that PADP can significantly improve the I/O performance of hybrid PFSs.

Keywords: Parallel I/O System, Parallel File system, Solid State Drive.

1 Introduction

Data I/O access is a key performance bottleneck of modern computer systems. To tackle this problem, parallel file systems (PFS) have been proposed to speed up large-scale data accesses. In many PFSs (e.g., OrangeFS [1] and Lustre [2]), a file usually is distributed across multiple file servers with a fixed-size stripe. When serving a client request concurrently by multiple nodes, the I/O bandwidth is significantly aggregated and improved. However, while PFSs favor large requests, they fail to perform well when serving clusters of small requests, especially random requests. Therefore, how to optimize PFS performance with different I/O patterns is still a challenging task the high performance computing (HPC) community is facing.

At the same time, newly emerged storage technologies, such as flash-based solid state drives (SSD), provide a new opportunity for I/O system design. Compared to traditional HDDs, SSDs have higher storage density, lower power consumption, a smaller thermal footprint and orders of magnitude higher performance [3]. However, it is not practical to replace HDDs completely with SSDs in a large-scale HPC system for two reasons. First, building PFSs solely based on SSDs may be too expensive for most systems. Second, HDDs have several advantages that satisfy the demands of HPC well, such as large capacity, attractive cost per storage unit, and decent peak bandwidth for large requests. Therefore, hybrid PFSs that consists of HDD-based file servers (HServer) and SSD-based file servers (SServer), provide a promising solution for data-intensive applications [4, 5]. This kind of situation is especially common in low-cost clusters, where cost is a critical issue and old components have to be used as much as possible.

Although hybrid PFSs have brought new opportunities to many application domains, there are many technical challenges yet to be solved before it is widely adopted. One of the major issues is, with the limited SServer resources, how to maximize the overall I/O system performance. Optimizing file system data placement (layout) is one of the most effective way to reach this goal. Researchers have made significant efforts to get an optimal data placement via adjusting the file stripe size [6, 7], or file stripe distribution method [8]. All these techniques introduce an optimal data layout based on application's data access patterns and guarantee the load-balance of all file servers. However, these approaches often focus on homogeneous PFSs, there is little consideration toward the performance difference among heterogeneous file servers in hybrid PFSs. For instance, for the file stripe size adjustment problem addressed in [6–9], the solutions are based on the assumption that all of the file servers are based on traditional HDDs.

However, in hybrid PFSs, the performance of each kind of file servers (i.e., HServer and SServer) varies significantly. A high-speed SServer can finish processing and storing data in a local SSD of the server faster than the relatively low-speed HServer. Traditional PFSs usually place a large file across multiple file servers with a fixed-size stripe. These data placement schemes are suitable for homogeneous environments, because they are able to provide concurrent I/O accesses and good load balance among multiple file servers. When applied to hybrid PFSs where file servers are not identical fixed-size stripe placement schemes may lead to severe load-imbalance among file servers and can significantly degrade the I/O performance. As high-speed SServers spend plenty of time on waiting the slower HServers during the I/O service, the potential of the hybrid PFSs is not fully utilized.

In this paper, we propose a performance-aware data placement scheme (PADP) to optimize the data placement in hybrid PFSs. Compared to traditional placement schemes, PADP distributes file data on different file servers with appropriate varied-size stripes according to their storage performance. For heterogeneous system it is not easy to determine the appropriate stripe sizes for different file servers. There are three main reasons. First, the performance of file server can be impacted significantly by I/O patterns. Second, even under the same I/O patterns, the performance between HServer and SServer can be different due to their distinct storage media characteristics. Third, besides the storage cost, the overall I/O performance is a function of the underlying network, which should be considered when evaluating the data access performance. The proposed performance-aware scheme takes the above three challenges into consideration in the data placement on hybrid PFSs, and can minimize the overall I/O access time from a client point of view. In addition, it can be extended to systems with other kinds of heterogeneous file servers, system configurations, and I/O patterns.

Specifically, we make the following contributions.

– We develop an analytical model to evaluate the overall I/O completion time of each data access in hybrid PFSs.
– Based on the cost model, we use a linear programing method to determine the optimal stripe size for each file server.
– We propose a performance-aware data placement scheme with the optimal stripe sizes to improve the hybrid file system performance.
– We implemented a prototype of PADP under OrangeFS, and evaluated its performance with IOR benchmark. Extensive experimental results show that PADP can significantly improve the I/O throughput of hybrid parallel file systems.

The rest of this paper is organized as follows. Section 2 discusses the related work. The design and implementation of PADP is described in section 3. Section 4 presents the performance evaluation with commonly used benchmark. Finally, conclusions are summarized in section 5.

2 Related Work

Optimizing data placement of parallel file system is an effective approach to improve I/O performance. Parallel file systems usually provide several data placement policies for different I/O workloads [8], such as simple stripe, two dimensional stripe, and variable stripe. Data partition [10,11] and replication [8,12] techniques are also widely used to optimize data placement on file servers consistent with I/O workloads. Because data accesses for some scientific applications usually show several regular patterns [13], some data placement optimization techniques rely on the prior knowledge of data access patterns [9].

For applications that access I/O systems non-uniformly, simple stripe placement schemes are not able to obtain high performance. Segment-level placement scheme logically divides a file into several segments such that an optimal stripe size is assigned for each segment with non-uniform access patterns [6]. Server-level adaptive placement strategies adopt different stripe sizes on different file servers to improve the overall I/O performance of parallel file systems [7]. However, this work is not suitable for systems built on heterogeneous file servers. AdaptRaid addresses the load imbalance issue in heterogeneous disk array by optimizing data distribution with adaptive number of blocks [14]. However, it aims to reduce I/O latency rather than improving I/O bandwidth, and needs not to consider the network cost in data accesses.

Because SSDs exhibit obvious performance benefits over traditional HDDs, they are commonly integrated into parallel file system to improve I/O performance. Currently, most SSDs serve as a cache to traditional HDDs [15,16] or persistent storage of file data [17,18]. Most of these techniques, however, are done on a single file server. Our previous work CARL [4] selectively places file regions with high access costs onto the SSD-based file servers at the I/O middleware level.

All the aforementioned data placement techniques are effective in improving the performance of parallel file systems. However, there is little effort devoted on data placement in a hybrid parallel file system configured with HServers and SServers.

3 Design and Implementation

3.1 The Basic Idea of PADP

The proposed data placement scheme, PADP, aims to optimize the file data placement on heterogeneous file servers with varied-size stripes based on their storage performance. Figure 1 shows the idea of PADP. Similar to traditional data placement method, PADP dispatches the file data across file servers in a round-robin fashion, but the high-performance SServers are expected to store and process larger file stripes compared with low-performance HServers, so that all the file servers can complete processing their I/O requests within about the same time.

As we have mentioned previously, determining the appropriate stripe sizes on heterogeneous file servers is not an easy task due to three reasons. First, the file server performance can be impacted significantly by I/O patterns, such as request size, I/O operation (read or write), number of processes, etc. Second, the server performance is also related with their storage media characteristics. HServer and SServer have different performance behaviors even under the same I/O patterns. Finally, besides the storage cost, the overall I/O performance is a function of the underlying network, which should be considered when evaluating the data access performance. In order to address these issues, we first propose an analytical model to evaluate the access time of file requests. Then we use a linear programming method to determine the appropriate stripe size for each file server. Finally we describe the performance-aware data layout scheme.

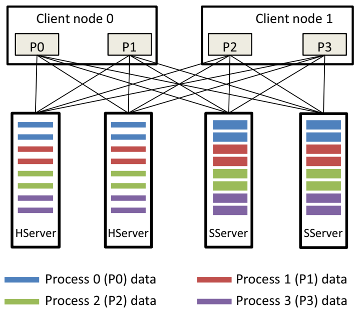


Fig. 1. Performance-aware data placement with varied-size stripes

3.2 Data Access Cost Analysis

Table 1. Parameters in cost analysis model

Symbol	**Meaning**
p	Number of client processes
c	Number of processes on one I/O client node
m	Number of HServers
n	Number of SSServers
h	Stripe size on HServer
s	Stripe size on SServer
r	Data size of one request
e	Cost of single network connection establishing
t	Network transmission cost of one unit of data
α_h	Startup time of one I/O operation on HServer
β_h	HDD transfer time per unit data
α_s	Startup time of one I/O operation on SServer
β_s	SSD transfer time per unit data

The cost is defined as the overall I/O time of each data access in hybrid PFSs. Table 1 lists the related parameters. Compared with previous work [6], this model is designed for heterogeneous environments. Please note that parameters for different types of requests on different storage media are differentiated when measuring the I/O time. The startup and transfer time are different between HServer and SServer. Generally, α_S is far smaller than α_H, and β_S is far greater than β_H because SSDs have no mechanical components. In addition, both α_H and α_S can be different between random and sequential operations, as we discuss in our experiments. Finally, while β_H is the same for reads and writes, β_S is different for them because writes on SSDs lead to background activities like garbage collection and wear leveling.

Before introducing the details of the model, we make following reasonable assumptions. First, all client nodes are separated from file servers in the system, which implies every data access involves network transmission. Second, the application-level parallel operations in each node are handled serially at the hardware layer, such as multiple network connections and storage accesses on file servers. Third, each I/O request involves all file servers, so that all servers can contribute to the aggregated I/O bandwidth. Assuming the stripe size of HServer and SServer is h and s respectively, the size of the data access is r, then

$$m \times h + n \times s = r \tag{1}$$

The data access cost mainly includes two parts: the network transmission time, T_{NET}, and the storage access time, T_{STOR}. Generally, T_{NET} consists of

T_E and T_X. T_E is the network connection for data transmission, T_X is the data transfer time on network. T_{STOR} consists of T_S and T_T, the former is the startup time, and the latter is the actual data operation (i.e., read/write) time on storage media. Thus the cost of one data access can be described as follows.

$$T = T_E + T_X + T_S + T_T \tag{2}$$

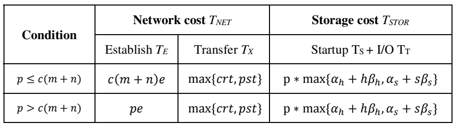
T_E is determined by the number of establishing connections to each file server. As each file server is accessed by p processes and the p network connections have to be established serially, $T_E = pe$. T_X is determined by the amount of data accessed on each file server. For HServer, $T_X = pht$; for SServer, $T_X = pst$. In a parallel environment, the overall network transfer time is the maximum of all servers, thus $T_X = max\{pht, pst\} = pst$. On the other hand, each client node needs to establish network connections and transfer their data from all file servers serially, thus $T_E = c(m+n)e$ and $T_X = crt$. As network connections are affected by both file servers and client nodes, the network establish time T_E and network transfer time T_X are chosen in a prudential way when they are different at client nodes and file servers. If the number of network connections on client nodes is larger than that of file servers ($c(m+n) > p$), the number of client connections $c(m+n)$ is used. That is

$$T_E + T_X = \begin{cases} c(m+n)e + \max\{crt, pst\}, & c > \frac{p}{m+n} \\ pe + \max\{crt, pst\}, & \text{otherwise} \end{cases} \tag{3}$$

The startup time T_S and data transfer time T_T of each file server is only determined by the number of sequential I/O operations, namely the number of client processes assigned on that server. For HServer, $T_S = p\alpha_h, T_T = ph\beta_h$; for SServer, $T_S = p\alpha_s, T_T = ps\beta_s$. In a parallel environment, the storage cost T_{STOR} is determined by the maximal storage cost of all servers. Thus

$$T_S + T_T = p \times \max\{\alpha_h + h\beta_h, \alpha_s + s\beta_s\} \tag{4}$$

Based on Equation 3 and 4, the overall cost values of each data access are shown in Figure 2. This cost model provides a detailed analysis of completion time for data accesses in hybrid PFSs. Although there are several parameters in the model, for most applications, the runtime variables such as c, p, m and n are fixed for each run. In general, for a given system, e, t, α and β can be regarded as constants.



Condition	Network cost T_{NET}		Storage cost T_{STOR}
	Establish T_E	Transfer T_X	Startup T_S + I/O T_T
$p \le c(m+n)$	$c(m+n)e$	$\max\{crt, pst\}$	$p * \max\{\alpha_h + h\beta_h, \alpha_s + s\beta_s\}$
$p > c(m+n)$	pe	$\max\{crt, pst\}$	$p * \max\{\alpha_h + h\beta_h, \alpha_s + s\beta_s\}$

Fig. 2. Cost formulas for hybrid PFSs

3.3 Determining the Optimal Stripe Sizes for Each File Server

Figure 2 shows that the data access cost T can be significantly impacted by the file server stripe sizes h and s. In other words, data placements with different stripe sizes lead to substantially variable access cost. In order to get the optimal I/O performance, the proposed data placement will find suitable stripe sizes to minimize the data access cost in hybrid PFSs. Thus, the optimization problem can be described as minimizing function F described in Equation 5 while satisfying the size constraints described in Equation 1.

$$F = \max\{crt, pst\} + p \times \max\{\alpha_h + h\beta_h, \alpha_s + s\beta_s\} \tag{5}$$

According to the member values in the two maximum functions in Equation 5, such problem can be translated into four linear programming (LP) problems with two unknown variables representing the stripe size h and s. The final problem is to choose the values of h and s so as to minimize F as below.

Case 1:

$$\text{Minimize} \qquad F = crt + ph\beta_h \tag{6}$$

$$\text{s.t.} \begin{cases} mh + ns & = r \\ ps & \le cr \\ \alpha_s + s\beta_s & \le \alpha_h + h\beta_h \end{cases} \tag{7}$$

Case 2:

$$\text{Minimize} \qquad F = crt + ps\beta_s \tag{8}$$

$$\text{s.t.} \begin{cases} mh + ns & = r \\ ps & \le cr \\ \alpha_h + h\beta_h & \le \alpha_s + s\beta_s \end{cases} \tag{9}$$

Case 3:

$$\text{Minimize} \qquad F = pst + ph\beta_h \tag{10}$$

$$\text{s.t.} \begin{cases} mh + ns & = r \\ cr & \le ps \\ \alpha_s + s\beta_s & \le \alpha_h + h\beta_h \end{cases} \tag{11}$$

Case 4:

$$\text{Minimize} \qquad F = pst + ps\beta_s \tag{12}$$

$$\text{s.t.} \begin{cases} mh + ns & = r \\ cr & \le ps \\ \alpha_h + h\beta_h & \le \alpha_s + s\beta_s \end{cases} \tag{13}$$

The final stripe sizes of h and s are determined by the case where the objective function F achieve the smallest value among the four cases. Please note that the optimal h can be zero, which means placing file data only on the underlying SServers leads to better performance. As the linear program is expressed with two unknown variables, the search space is very small and solving the program requires acceptable time cost.

3.4 Performance-Aware Data Placement Scheme

Based on the optimal stripe sizes h and s, PADP is able to achieve the optimal file data placement for data-intensive applications. This approach requires a prior knowledge of data access patterns of applications. As described in [9, 10], many HPC applications access their files with either regular data access patterns or predictable behaviors. For example, numerous tools were developed to trace I/O requests for these applications [13]. These applications are often executed on a computer cluster many times, and the file access patterns are generally independent of the data values stored. The request patterns can be learned from previous runs. Figure 3 shows the procedure of the optimal data placement scheme. Basically, the proposed data placement scheme consists of three phases: pre-estimation, layout determination, and data placement. In the pre-estimation phase, the related parameters in the cost model are estimated. As described previously, the network parameters, such as e and t, the storage parameters, such as α_h, β_h, α_s, β_s, and the system configuration parameters, such as m and n , can be regarded as constants. In the layout determination phase, the cost model and the linear programing method are used to calculate the optimal file stripe sizes h and s for HServers and SServers. In this phase, the applications access patterns, such as c, p, r are used as inputs. Determining the optimal stripe sizes is relatively fast since it requires solving only a small two-variable linear programming problem. In the data placement phase, the optimized file stripes can be used for the file data distribution for the applications, either by creating new files for later runs of the applications, or adjusting the file layout by file copy operations in the existing parallel file systems.

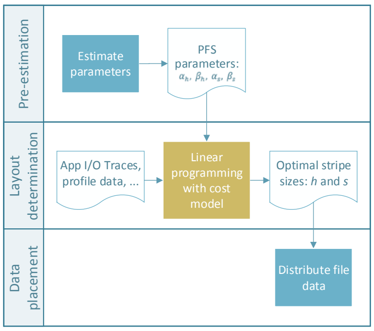

Pre-estimation
Estimate parameters
PFS parameters: α_h, β_h, α_s, β_s
Layout determination
App I/O Traces, profile data, ...
Linear programming with cost model
Optimal stripe sizes: h and s
Data placement
Distribute file data

Fig. 3. The procedure of the performance-aware data placement scheme

3.5 Implementation

We have implemented a prototype of the performance-aware data placement scheme in OrangeFS.

Pre-estimation. We use one file server in the parallel file system to test the startup time α and data transfer time β for HServers and SServers with sequential/random and read/write patterns. Please note that the parameters can vary with different I/O patterns. In addition, we use a pair of nodes (one client node and one file server) to estimate network parameters, the network connection establishing time e and network transfer time t. We repeat the tests with thousands of times (the number is configurable), and then calculate their average values, which are used as the parameter values.

Optimal Data Distribution. Once obtaining the optimal stripe sizes for HServers and SServers, we use them to distribute file data among available file servers for better I/O performance. The OrangeFS file system supports an API for implementing specific variable stripe distribution by default. The variable stripe distribution is similar to simple stripe, except that the stripe size can be configured to be different on different file servers. In OrangeFS, parallel files can either be accessed by the direct PVFS2 interface or the POSIX interface. When using the direct PVFS2 interface, we utilize the "pvfs2-xattr" command to set the data distribution of directories where the application files are located. In addition, when a new file is created, we use the "pvfs2-touch" command with the "-l" option to specify the order of the file servers, so that the file stripe size h and s can be configured for the corresponding HServers and SServers accordingly.

3.6 Discussion

One concern of PADP is that it can potentially lead to more storage space consumption for SServers, which might perhaps be an unwanted feature by users. Fortunately, most file systems do not make full use of the storage space in the underlying devices. In practical system, this issue is not frequently encountered if the SSD space is enough. In the worst case, with the possibility of an SServer running out of its space, we design a data migration method to balance the storage space by moving data from SServers to HServers, so that the available remaining space on SServers can be guaranteed for new coming requests. This problem can also be addressed by using the hybrid PFS to store performance-critical data (e.g. frequently accessed data) and the PFS only on HServers to store the rest of the data.

4 Performance Evaluation

In this section, we evaluate the performance of the proposed data placement scheme through several benchmark-driven experiments.

4.1 Experimental Setup

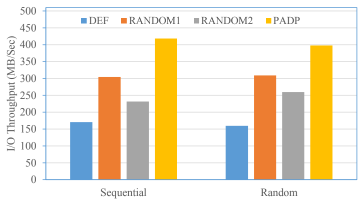
We conducted the experiments on a 65-node SUN Fire Linux cluster, where each node has two AMD Opteron(tm) processors, 8GB memory and a 250GB HDD. 16 nodes are equipped with additional OCZ-REVODRIVE 100GB SSD. All nodes are equipped with Gigabit Ethernet interconnection. The parallel file system is OrangeFS 2.8.6. Among the available nodes, we select eight nodes as client computing nodes, eight nodes as HServers, and eight nodes as SServers. By default, the hybrid OrangeFS file system is built on six HServers and two SServers.

We compare three data placement schemes: the default scheme (DEF), the random scheme (RANDOM), and the proposed PADP scheme. In DEF, the file data is placed across all file servers with a fixed-size stripe of 64KB; in RANDOM, the file stripe sizes are randomly selected. To be simple, the stripe size pair $< h, s >$ is used in the following sections, which means the stipe sizes on HServers and SServers are h and s respectively. The popular benchmark IOR [19] is used to test the performance.

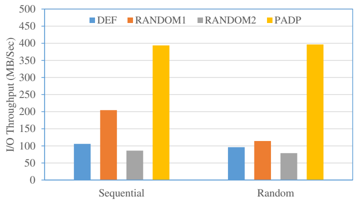
4.2 IOR Benchmark

Performance with Different Read Write Modes. Unless otherwise specified, the IOR benchmark runs with 8 processes, each of which performs I/O operations in individual mode on a 10GB shared file. The request size is kept to 512KB. Figure 4 demonstrates the I/O performance of IOR with sequential and random I/O access mode under the three data placement schemes. In the figure, the randomly selected stripe size pair is <32KB, 96KB> in RANDOM1, and <96KB, 32KB> in RANDOM2. For PADP, the optimal stripe sizes for sequential and random read, sequential and random write, are <28KB, 100KB>, <20KB, 108KB>, <24KB, 104KB>, and <36KB, 92KB> respectively. From the results we can observe that PADP has the best performance of all schemes. By using the optimal stripe sizes for HServers and SServers, PADP can improve read performance by up to 149.2% over DEF with all I/O access modes, and write performance by up to 271.8%. Compared with RANDOM1 and RANDOM2, PADP can improve the read performance by up to 80.6% and write performance by up to 357.1% for all I/O access modes. This shows that the idea of PADP works well and the stripe size determining formula of PADP is effective.

In order to give a detailed explanation, Figure 5 plots the I/O time of each file server during a 10-second IOR execution period when IOR performs sequential read operations under the three schemes. The I/O time is normalized to that of the minimum I/O time of all file servers. Among the eight file servers, server 0 to 5 are HServers, and the rest are SServers. From Figure 5, we can observe that the I/O loads of HServers and SServers are severely skewed under scheme DEF and RANDOM. In contrast, the optimal data placement scheme PADP can significantly eliminate the load imbalance among file servers. Thus, PADP improves the file system performance.



(a) Read throughput



(b) Write throughput

Fig. 4. Throughputs of IOR under different placement schemes with different I/O modes

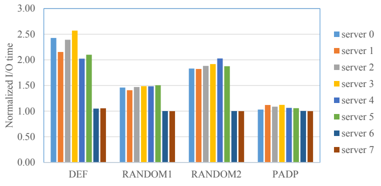
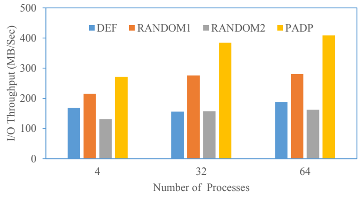
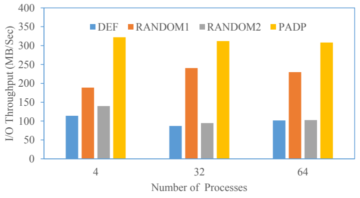


Fig. 5. I/O time on each node under different data placement schemes

Performance with Different Number of Processes. The I/O performance is also evaluated with different number of processes. The IOR benchmark is executed under the random access mode with 4, 32 and 64 processes. In this test, RANDOM1 and RANDOM2 use the same stripe size configuration as previous test. As show in Figure 6, the results are similar to the previous test. PADP has the best performance among the three schemes. Compared with DEF, PADP improves the read performance by 60.8 %, 146.3%, and 118.4%z respectively with 4, 32 and 64 processes, and write performance by 182.3%, 257.8 %, and 202.7%. Compared with RANDOM, PADP can brings a read performance improvement by up to 107.9%, 145.2%, and 151.6% respectively with 4, 32 and 64 processes, and write performance improvement by up to 130.3%, 228.8%, and 200.3%. These results show that PADP has very good scalability with the number of I/O processes.



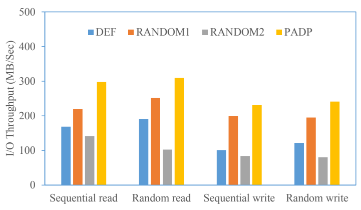
(a) Read throughput



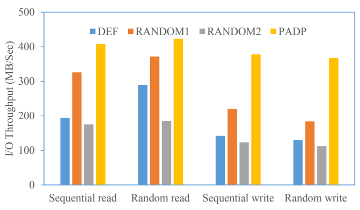
(b) Write throughput

Fig. 6. Throughputs of IOR with varied number of processes

Performance with Different Request Sizes. Figure 7 demonstrates the I/O performance of IOR with request size of 128KB and 2048KB. The number of processes is fixed to 16, and IOR issues random requests. Figure 7(a) shows the result for the relatively small requests. We can observe that PADP can improve the read performance by up to 76.3%, and write performance by up to 127.9% in comparison with the default data placement scheme DEF. Compared with RANDOM, PADP also has better performance: the read performance is increased by up to 201.7 %, and write performance is increased by up to 199.4%. When the request size is 128KB, it is worth noting that the optimal stripe sizes in PADP are <0KB, 32KB> for all I/O modes. This implies that distributing the file only on the four SServers leads to the highest I/O performance if the requests are relatively small. For larger size 2048KB, PADP distributes the file across both HServers and SServers to achieve optimal performance, as shown in Figure 7(b). This is because all servers are working cooperatively and this can lead to better I/O performance for large requests. These results show that PADP has a good scalability for different request sizes.



(a) Request size is 128K



(b) Request size is 2048K

Fig. 7. Throughputs of IOR with varied request sizes

Performance with Different Server Configurations. The I/O performance is examined with varied ratios of SServers to HServers. The OrangeFS is built using HServers and SServers with the ratios of 5:3, and 3:5. Figure 8 shows the average I/O bandwidth with different file server configurations. As it can be seen from the results, PADP can improve I/O throughput for both data read and write. When the ratio is 5:3, the randomly selected stripe sizes of RANDOM1 and RANDOM2 are <34KB, 114KB> and <82KB, 34KB> respectively. Compared with DEP, the read performance improves by up to 100.9% , and write performance improves by up to 154.1%. We can observe that PADP can increase the read performance by up to 105.9%, and write performance by up to 169.6% the RANDOM scheme. When the ratio is 3:5, the randomly selected stripe sizes of RANDOM1 and RANDOM2 are <29KB, 85KB> and <89KB, 49KB> respectively. We can observe that PADP has the similar behavior. In the experiments, read and write performance improved as the number of SServers increased. This is because the I/O performance of SServers is efficiently utilized by PADP. By using the optimal stripe sizes determined by the linear programing method in

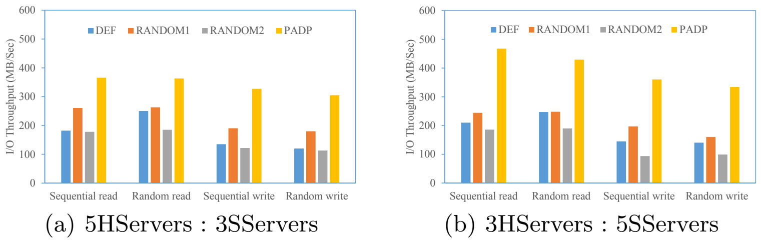


Fig. 8. Throughputs of IOR with varied file server configurations

this paper, PADP can significantly improve the hybrid file system performance with all file server configurations.

5 Conclusions

In this study, we have proposed a performance-aware data placement (PADP) scheme, which distributes data across HDD and SSD file servers with adaptive stripe sizes based on their storage performance. We have presented the proposed PADP data placement optimization scheme and implemented it in the OrangeFS file system. Essentially, PADP provides a better matching of data access characteristics of an application with the storage capabilities in the file servers of the underlying heterogeneous file system. Experimental results of representative benchmark show that PADP can significantly improve the file system performance.

References

1. Orange File System, `http://www.orangefs.org/`
2. Microsystems, S.: Lustre File System: High-performance Storage Architecture and Scalable Cluster File System. Tech. Rep. Lustre File System White Paper (2007)
3. Chen, F., Koufaty, D.A., Zhang, X.: Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives. In: Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, pp. 181–192 (2009)
4. He, S., Sun, X.-H., Feng, B., Huang, X., Feng, K.: A Cost-Aware Region-Level Data Placement Scheme for Hybrid Parallel I/O Systems. In: Proceedings of the IEEE International Conference on Cluster Computing (2013)
5. He, S., Sun, X.-H., Feng, B.: S4D-Cache: Smart Selective SSD Cache for Parallel I/O Systems. In: Proceedings of the International Conference on Distributed Computing Systems (2014)
6. Song, H., Yin, Y., Sun, X.-H., Thakur, R., Lang, S.: A Segment-Level Adaptive Data Layout Scheme for Improved Load Balance in Parallel File Systems. In: Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 414–423 (2011)

7. Song, H., Jin, H., He, J., Sun, X.-H., Thakur, R.: A Server-Level Adaptive Data Layout Strategy for Parallel File Systems. In: Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum (2103)
8. Song, H., Yin, Y., Chen, Y., Sun, X.-H.: A Cost-Intelligent Application-Specific Data Layout Scheme for Parallel File Systems. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing, pp. 37–48 (2011)
9. Yin, Y., Li, J., He, J., Sun, X.-H., Thakur, R.: Pattern-Direct and Layout-Aware Replication Scheme for Parallel I/O Systems. In: Proceedings of 27th IEEE International Parallel and Distributed Processing Symposium (2013)
10. Wang, Y., Kaeli, D.: Profile-Guided I/O Partitioning. In: Proceedings of the 17th Annual International Conference on Supercomputing, pp. 252–260. ACM, San Francisco (2003)
11. Rubin, S., Bodik, R., Chilimbi, T.: An Efficient Profile-Analysis Framework for Data-Layout Optimizations. ACM SIGPLAN Notices 37(1), 140–153 (2002)
12. Huang, H., Hung, W., Shin, K.G.: FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption. In: Proceedings of the 20th ACM Symposium on Operating Systems Principles, pp. 263–276. ACM, New York (2005)
13. Yin, Y., Byna, S., Song, H., Sun, X.-H., Thakur, R.: Boosting Application-Specific Parallel I/O Optimization Using IOSIG. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 196–203 (2012)
14. Cortes, T., Labarta, J.: Taking Advantage of Heterogeneity in Disk Arrays. Journal of Parallel and Distributed Computing 63(4), 448–464 (2003)
15. Pritchett, T., Thottethodi, M.: SieveStore: a Highly-Selective, Ensemble-level Disk Cache for Cost-Performance. In: Proceedings of the 37th Annual International Symposium on Computer Architecture, pp. 163–174 (2010)
16. Zhang, X., Liu, K., Davis, K., Jiang, S.: iBridge: Improving Unaligned Parallel File Access with Solid-State Drives. In: Proceedings of 27th IEEE International Parallel and Distributed Processing Symposium (2013)
17. Yang, Q., Ren, J.: I-CASH: Intelligently Coupled Array of SSD and HDD. In: Proceedings of the IEEE 17th International Symposium on High Performance-Computer Architecture, pp. 278–289 (2011)
18. Chen, F., Koufaty, D.A., Zhang, X.: Hystor: Making the Best Use of Solid State Drives in High Performance Storage Systems. In: Proceedings of the International Conference on Supercomputing, pp. 22–32 (2011)
19. Interleaved Or Random (IOR) Benchmarks (2014), `http://sourceforge.net/projects/ior-sio/`

Security Analysis and Protection Based on Smali Injection for Android Applications

Junfeng Xu, Shoupeng Li, and Tao Zhang

China Information Technology Security Evaluation Center
Beijing, China, 100085
{xujf,lisp,zhangt}@itsec.gov.cn

Abstract. Smali code and .Dex file can be completely compiled and decompiled reciprocally. Thus any new functions can be injected into an existing android application directly after decompiling it into smali code under the condition of that we needn't to modify any java code to develop the application. This leads the android applications to be modified and cracked arbitrarily. In order to prevent it from being decompiled and bundled malicious code, we summarized current typical methods of anti-crack and anti-decompilation, then propose two new solutions based on smali injection, which can protect the security of the android applications effectively.

Keywords: Android Security, Smali Injection, Software Protection.

1 Introduction

Due to its good user experience, excellent architectural design, convenient way of software development as well as the company behind the powerful technical support, the Android system get the favour of a large number of users and software developers. Its global market share has reached 74% in the first quarter of 2013. However, because of the openness of the system, a growing number of hackers and criminals through the induction of users to install malicious software, such as stealing a large number of users' privacy information, and then make a lot of illegal profits. According to wooyun vulnerability database statistics, as of February 14, 2014, domestic Android application vulnerabilities have been found to 229, growth trend is obvious.

Due to the reverse analysis for the Android application can be decompiled, the developer can reverse changes of target application code, malicious code and binding, and the reverse code after secondary packaging and signature. This leads to more and more malicious to be spread easily. Therefore, this paper puts forward some protection mechanisms for android applications, which can effectively protect android applications avoiding been cracked.

The security of the Android mobile phone has received widely attention. Many researchers related research on the security of the Android was carried out. Compared with representative William Enck et al. Development of Kirin system [1], it summed up some of the Android platform Permission in the level of safety rules, and is used to detect whether there is a danger of Android application target Permission configuration. William Enck also designed a dynamic stain tracking system for the Android platform TaintDroid [2], which is used to detect the privacy information disclosure problem in Android applications. In addition, addressing static analysis on the application of the Android, William Enck has also carried on effective attempt to use the ready-made Fortify analyzes the Android Java source code [3]. Adrienne Porter Felt and others Permission mechanism carried out in-depth analysis on the application of the Android. STOWAWAY, a simple tool, is developed to detect whether the Android to apply for the necessary Permission while installation [4]. Adrienne Porter Felt et al. also for the Permission of the Android related attack methods to carry out the study, found that Android malware could use the IPC privileges to complete the operation form of attack (that is Permission to Re-delegation attack), and discusses how to implement in the Android IPC mechanism corresponding defense [5]. In terms of the Android security protection, Machigar Ongtang and others developed an Android security enhancement frame system Sanit [6], the Android Permission strategy expanded, can support more kinds of installation and runtime Permission strategy. Michael Dietz et al. developed a lightweight Quire the Android security defense system [7], the main track of the IPC chain analysis and lightweight signature verification.

In [8] DroidChameleon, a systematic framework with various transformation techniques, is developed to anti-malware products for Android and test how resistant they are against various common obfuscation techniques. In [9], the proposed methodology relies on the repackaging of a compiled application and the injection of a reporter at byte code level. Thus, such a methodology enables the user to audit suspicious applications that ask permissions to access private data and to know if such an access has occurred. [10] prevented these exploits by modifying Androids Intent handling behavior to err on the side of safety except where the developer seems to explicitly specify[11] In this work the author present a protection system that resides on the mobile phone. the solution works by partitioning the phone software stack into the application operating system and the communication partition.

Addressing decompilation and Smali injection, this paper summarizes the general security issues on android applications . Especially, it describes the security features of smali injection for android applications and gives an example to reveal the smali injection technologies. After lots of experiments, we find the security vulnerabilities in the decompilers. Then we propose two solutions to avoid the android applications to be decompiled into smali code and Java code. In the end, we conduct two experiments to test the proposed solutions. The results reveal that the solutions work effectively.

The remainder of this paper is organized as follows. In Section, 2, the backgroud about android applications are described. Section presents how to decompile an android application and inject a malware into it. Section summarizes some typical mechanisms to anti-decompilation based smali injection, proposed two solutions for it. Finally, the concluding remarks are given in Section 6.

2 Background

2.1 The Harmfulness of Android Malware

In the first quarter of 2013 global market share is much higher than the IOS Android operating system, Windows Phone, reached 64.2%, along with the wide use of the Android platform, Android platform malicious sample size also increased dramatically, especially new malicious sample size only in the first half of 2013, more than 2012 the number of all the year round. Consumption rates, privacy, stealing, malicious deduction, cajoles fraud and malicious behavior of hooliganism is still by far the most common.

Consumption rates: in the case of user knowledge or unauthorized, by automatically dial the phone, send SMS, MMS, email, frequently to connect to the Internet, lead to the loss of user charges.

Privacy steal: in the case of user knowledge or unauthorized, get SMS/MMS content, access email, access the address book content, get phone records, obtain call content, location information and access to other users' personal information, etc.

Malicious deduction: users don't even know or unauthorized cases, users don't even know or without authorization, through hidden execution, cheat users click, order all kinds of charging or using mobile payment, economic losses lead to the user's behavior.

Lure fraud: users don't even know or without authorization, through forged, tampered with, hijacked SMS, MMS, email, address book, call records, favorites, desktop, tricking users, and improper purpose, etc.

Rogue behavior: in the case of user knowledge or unauthorized, long memory system, automatic bundling unknown third party software installed, automatically add, modify, delete, favorites, shortcut, pop-up ads, window, etc.

2.2 Reverse Analysis

Reverse analysis process is an analysis of the target system, its purpose is to identify the components of the system and the relationship between them, and in the form of other or on a higher level of abstraction, characterization of reconstruction system. Software reverse analysis can also be regarded as "retrograde" of the development cycle. Under this definition, to reverse analysis, a software program that is similar to the retrograde development steps in traditional waterfall model, namely the implementation phase of the output VAT back at the design stage of conception. Software reverse engineering is only a test or analysis of the process, it will not change the target system.

William Enck etc to decompiled Dalvik bytecode in on the source of data flow analysis and control flow analysis, structure analysis and semantic analysis, this method is mainly in the source code level security analysis, because the disassembly and confusing technology, make the source and the bytecode is distinct, and the impact analysis of results.

Study of Android applications bytecode file first operation is to reverse the bytecode file, get the Dalvik bytecode information. Now to reverse the Android application tool is more, more commonly used include Apktool, Baksmali, AXMLPrinter2, IDA pro, etc.

A. Schmidt and other implements A malware detection system, the system is based on client-server model, can undertake collaborative detection. With the method of static analysis of the Android executable file, in the Android environment with readelf command samples application function call information as the data detection of malicious software. The authors chose the less than 100 Linux command function invocation list of these functions and readelf extraction as a benign training set, and then download the 240 malware and extract function call list as malicious training samples, and then use a variety of classifying function call list classification algorithm, then the results of the classification and comparison of function call list of malicious software, to determine whether an application is malicious applications. It USES static analysis is analysis of the function call relationship, not for any flow sensitive analysis.

3 Decompile an Android Application and Inject a Malware into It

3.1 Decompile an Android Application

Assuming that my working folder is $AndroidDecompile, first of all, we copy system.img in several important odex files to the working directory, which are: core.odex, ext.odex, framework.odex, android.policy.odex, services.odex. In the $AndroidDecompile, download the following tools:

- **Baksmali:** http://code.google.com/p/smali/downloads/list
- **Smali:** http://code.google.com/p/smali/downloads/list
- **Dex2jar:** http://code.google.com/p/dex2jar/downloads/list
- **JD-GUI:** http://java.decompiler.free.fr/?q=jdgui
- **AutoSign:** http://d.download.csdn.net/down/2768910/
- **Apktool:** http://code.google.com/p/android-apktool/

Suppose we have an application, it compiled class files were took out separately, which are two file app.apk and app.odex, put them under the $AndroidDecompile.

1. Use baksmali.jar to divide odex files into smali files

```
$Java -jar baksmali-1.2.5.jar -x app.Odex
```

If successful, it will generate a out under $AndroidDecompile directory, there are some ".smali suffix" files.

2. Use the smali.jar in the out/smali directory to convert classes.Dex

```
$Java -Xmx512M -jar smali-1.2.5.jar out -o classes.Dex
```

Classes.Dex is Dalvik VM used by the compiled class file format, will be in the normal the apk file.

3. Use dex2jar to decompile classes.dex into a jar file

After download dex2jar package and decompression, there will be dex2jar. Sh (and dex2jar. Bat) file, if classes.dex file and dex2jar.sh in the same directory, use the following way to decompile classes.dex into jar files:

```
$dex2jar.sh classes.Dex
```

If it succeeds in the current directory to generate the compiled file classes.dex. Dex2jar.Jar. Dex2jar namely can dex file operation, also can directly operate the apk, its using rules as follows:

```
$dex2jar file1.apk file2.apk
```

4. Use the JD - GUI to view after decompiling jar file

JD - GUI is a visual Java decompiled code viewer. It can real-time to decompile class files into Java file for viewing. Unzip the download of the JD - GUI file, implement the JD - GUI executable files in directory, then load in the previous step the compiled classes.Dex.Dex2jar.Jar file.

5. From odex against the compiled classes.dex and other resource files repackaged into a complete apk

We assume that the above is the application of the compiled class file was spun off from the apk and the next thing to do is how to make the above steps in classes.dex to the rest of the apk file repackaged into a usable apk. Will first against the compiled classes.dex and the original app.apk (excluding classes.dex) to compress into a complete app.Apk (apk file compression tools available open). That is to say, the classes.Dex is put into the app.apk. Will download AutoSign file decompression, can see signapk.jar (there is a Sign.bat) file, execute the following command to app.Apk signature, you can generate the apk.

```
$Java -jar signapk.Jar testkey.pk8 app_signed.apk
```

3.2 Inject a Malware into an Android Application

Smali and baksmali are a compiler and a decompiler respectively from Google for the Dalvik virtual machine. It implements. Dex all functions. With smali and baksmali, the developer can implement a .Dex file for nondestructive compiling and decompiling. apktool can translated classes.Dex into smali directory. The directory is equivalent to the SRC directory under the Java project, and the . Smali files are linked to the SRC file. And the corresponding Java code is the same meaning, smali code of Java Code with. Dex executable file between the middle of the code. And smali code.Dex file can be completely intact Conversion, so directly after decompiling smali code embedded in a need to use the code, can not Changes under the condition of the original application functionality to add new functionality.

As described in Fig.1, in the process of decompilation, the developer can inject malware code into a normal android application. Then another malware android application is generated. But it looks like a normal application.

3.3 An Example of Smali Injection

In this example, we chose an android trial version application. If you want to use this program completely, you need to get the registration code of the program.

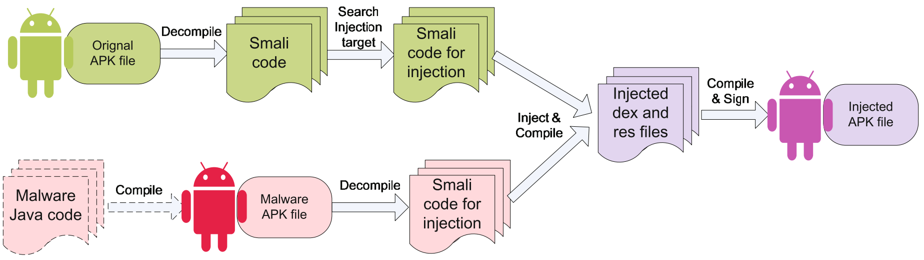
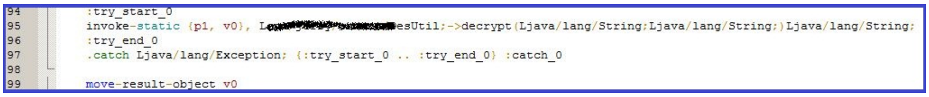


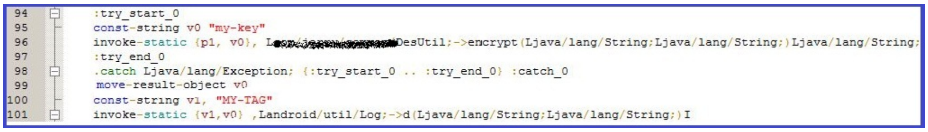
Fig. 1. Smali Injection process of android applications



```
    :try_start_0
    invoke-static {p1, v0}, L[redacted]esUtil;->decrypt(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
    :try_end_0
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_0} :catch_0

move-result-object v0
```

Fig. 2. Orignal Smali code of android applications



```
:try_start_0
const-string v0 "my-key"
invoke-static {p1, v0}, L[redacted]DesUtil;->encrypt(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
:try_end_0
.catch Ljava/lang/Exception; {:try_start_0 .. :try_end_0} :catch_0
 move-result-object v0
const-string v1, "MY-TAG"
invoke-static {v1,v0} ,Landroid/util/Log;->d(Ljava/lang/String;Ljava/lang/String;)I
```

Fig. 3. Injected Smali code of android applications

According to steps the above, we decompile the android application into smali code. after careful analysising the smali code, we found that the program in the test registration code, also want to decode the original string, which is obtained by decoding. So we can inject smali code, as shown in Fig. 2 and Fig. 3, run the decompiler after the registration code is generated. We can find the registration code using the command. Finally, we conduct two experiments to test the proposed solutions. The results reveal that the solutions work effectively.

```
$adb logcat | grep -i MY-TAG
```

4 Protection Based on Smali Injection for Android Applications

4.1 Code Confusion

Because Java bytecode is very easy to be decompiled, and general program developers have according to the function of classes and methods used in the habit of name, after the program was decompiled malicious code developers can easily from the class and the methods of function and modify this code contains. So in order to better protect the Java code, usually to code to confuse the class files. Through code confusion can be simply complicated, let developers malicious code is difficult to analyze the function of each method from the original program To modify the original program, in order to hide the implementation details of the program. Google to do this to add a code called ProGuard confusion software to the Android SDK, under the SDK/tool/ProGuard. In the new Android will generate a project under the root directory of the project. The property documents and proguard project. TXT file.

4.2 Dynamic Loading Class

By above knowable, even in the application code and joined the signature detection in confusion, malicious software after the decompiled by static scan to find it. So if you find a way to make the application to perform some unpredictable code, the malicious software will be hard to find and modify. Based on this idea we can easily think of can be used in Java to provide dynamic loading of a class to implement this scheme, unlike ordinary Java virtual machine, not Dalvik virtual machine directly by this kind of dynamic load, but by DexClassLoader and PathClassLoader two inherit from this class to the loadClass method

4.3 Jave Native Interface (JNI)

Due to the existence of the decompiled software, source code is almost open Android applications. Any layer in the Java implementation code may be malicious code writers find and tampered with. So if can improve the decompiled difficulty of implementation code, so the Android application security would improve greatly. With the Java language itself is not suitable for development of security software, so the Android provides us with an alternative, JNI calls. Although the Android application is generally done by the Java language, Android is an operating system based on the Linux kernel. All of the API eventually in the Android SDK is in Java layer through JNI access to the core library to achieve. Android also allows developers to define dynamic link library and in the Java layer through JNI access. This will bring us a new train of thought, dynamic link library is composed of C/C + +, decompiled difficulty greatly higher than that of Java code.

4.4 Signature Comparison Technology

Due to the fact that Android malware based on reverse engineering will change the program the signature, so you can by comparing the signature technology to prevent such attacks. By analyzing signapk.Jar source code can learn the whole process of Android application signature.

First, generate the MANIFEST. Manifest.mf. One by one to the application package other than the folder under the signature file generated SHA1 digital signature information. Then use Base64 coding algorithm for the digital signature information and will result in the MANIFEST. MF file. So, once the application package of files has been modified will produce different information, also can't be installed.

Second, generating cers.SF, file. Through the SHA1 - RSA algorithm and use private key of the MANIFEST.MF in information encryption. RSA is a kind of asymmetric encryption algorithm, the application of the signature file private key as a private key RSA algorithm, after installing a public key to decrypt it with the MANIFEST. The MF file information in contrast can draw after application package has been modified.

5 Our Solutions

5.1 Anti-dex2jar

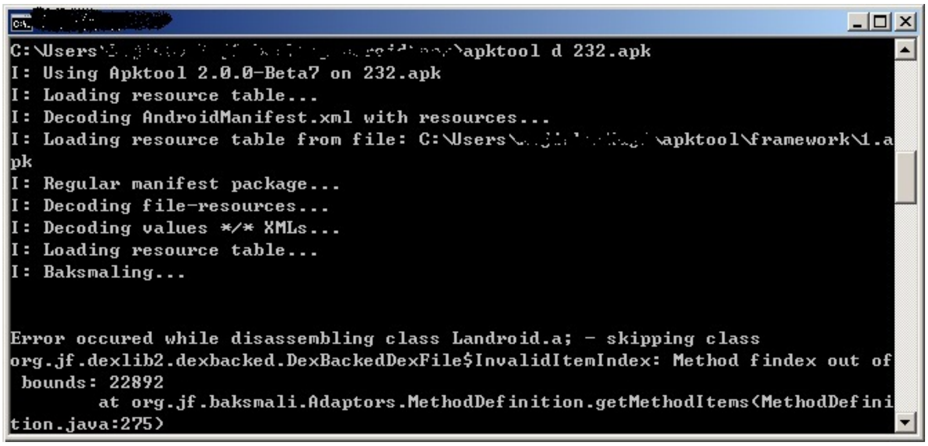
Discovering vulnerabilities of the decompiler(dex2jar) while processing apk or dex file, we can take advantage of it for anti-decompilation. After testing a large number of android applications, we find some of them which can't be decompiled. Then we try to analyze their characteristics of the java code of them. We have many apk download program and use the following command batch decompiled:

```
$for %%i in (*.apk) do apktool d %%i
```

As shown in Fig. 4, in the process of decompilation, errors occurred. The main reason is that dex2jar in analytic dex file while Dalvik doesn't support instructions. Through the analysis we found that the error of exception code is the method position() in class Sun.Security.Util.BitArray. So we can use this function when coding, as long as you call the position() method, which can make dex2jar error in the process of decompilation.

5.2 Anti-JD-GUI

Because smali code is hard to understand, the developer usually decompile smali code into Java code. With the analysis of the Java code, and then inject smali

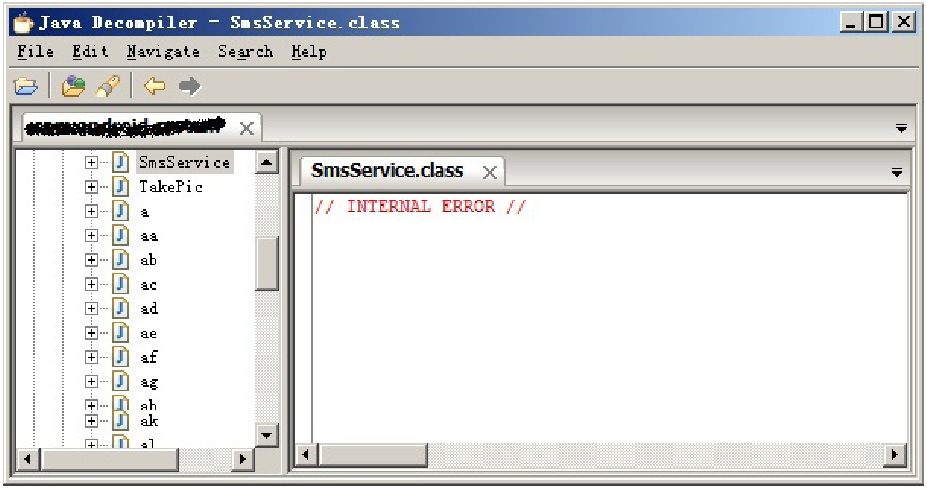


[redacted]

```
C:\Users\[redacted]\apktool d 232.apk
I: Using Apktool 2.0.0-Beta7 on 232.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\[redacted]\apktool\framework\1.a
pk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Loading resource table...
I: Baksmaling...


Error occured while disassembling class Landroid.a; - skipping class
org.jf.dexlib2.dexbacked.DexBackedDexFile$InvalidItemIndex: Method findex out of
 bounds: 22892
        at org.jf.baksmali.Adaptors.MethodDefinition.getMethodItems(MethodDefini
tion.java:275)
```

Fig. 4. Errors occur while decompiling smali code


Java Decompiler - SmsService.class
File Edit Navigate Search Help
[redacted]
SmsService
TakePic
a
aa
ab
ac
ad
ae
af
ag
ah
ak
al
SmsService.class

```
// INTERNAL ERROR //
```

Fig. 5. Errors occur while decompiling java code

code, modify the android applications. Some even directly modify Java code and programming into modified android applications. We prevent smali code from being decompiled into Java code. This will increase the difficulty of the injected directly into the smali code. Similar to the methods above, after we decompiled a lot of android applications, we found that there was an error in Java code. As shown in Fig. 5, after analysis, we found that if the methods in the program code length of more than 532 lines of code, this class cannot be decompiled into Java code. Thus, we can use this solution to prevent the smali code from being decompiled to java code.

6 Concluding Remarks

This paper investigate how to decompile an android application and inject a malware into it. Some typical mechanisms to anti-decompilation based smali injection are summarized. Addressing the smali injection for android application, we proposed two solutions. One prevents .dex file to be decompiled to smali code using dex2jar. The other can avoid smali code to be decompiled into java code. With the up-to-date decompilers are released, these solutions we proposed in the paper must be improved. Otherwise, the solutions will be no longer in force.

References

1. Enck, W., Ongtang, M., Mcdaniel, P.: On lightweight mobile phone application certification. In: ACM Conference on Computer and Communications Security, pp. 235–245. ACM (2009)

2. Enck, W., Cox, L.P., Jung, J., et al.: Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones, Vancouver, BC (October 2010)
3. Enck, W., Octeau, D., Mcdaniel, P., Chaudhuri, S.: A study of android application security. In: Proc. USENIX Security Symposium (2011)
4. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS (2011)
5. Felt, A.P., Hanna, S., Chin, E., Wang, H.J., Moshchuk, E.: Permission re-delegation: Attacks and defenses. In: 20th Usenix Security Symposium (2011)
6. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically rich application-centric security in android. In: Computer Security Applications Conference, ACSAC 2009, pp. 340–349 (December 2009)
7. Dietz, M., Shekhar, S., Wallach, D.S., Pisetsky, Y., Shu, A.: Quire: Lightweight provenance for smart phone operating systems, San Francisco, CA (August 2011)
8. Rastogi, V., Chen, Y., Jiang, X.: Catch me if you can: Evaluating android anti-malware against transformation attacks. IEEE Transactions on Information Forensics and Security 9, 99–108 (2014)
9. Berthome, P., Fecherolle, T., Guilloteau, N., Lalande, J.F.: Repackaging android applications for auditing access to private data. In: 2012 Seventh International Conference on Availability, Reliability and Security (ARES), pp. 388–396 (August 2012)
10. Cozzette, A., Lingel, K., Matsumoto, S., Ortlieb, O., Alexander, J., Betser, J., Florer, L., Kuenning, G., Nilles, J., Reiher, P.: Improving the security of android inter-component communication. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 808–811 (May 2013)
11. Mulliner, C., Liebergeld, S., Lange, M., Seifert, J.-P.: Taming mrhayes: Mitigating signaling based attacks on smartphones. In: 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 1–12 (June 2012)

A Novel Key Management Scheme in VANETs

Guihua Duan, Yun Xiao, Rui Ju, and Hong Song

School of Information Science and Engineering, Central South University, China
{duangh,songhong}@csu.edu.cn, xxy1220@yeah.net,
1584140885@qq.com

Abstract. To protect users' privacy and provide efficient and secure key management in Vehicular ad hoc networks (VANETs), this paper proposes a novel Efficient Dynamic Key Management scheme based on Dynamic Secret Sharing, namely EDKM-DSS. The proposed scheme replaces the complex group signature technology with the elliptic curve ElGamal threshold mechanism and the dynamic secret sharing technology, and realizes the key management of the group communication. Experiments results show that the proposed novel scheme can effectively improve the overall performance.

Keywords: VANETs, Key Management, Dynamic Secret Sharing, Elliptic Curve ElGamal Threshold Scheme.

1 Introduction

Vehicular ad hoc network (VANET) is an emerging new field and can provide better society interests. VANETs link vehicles traveling on the road together to form a unified network. This makes the drivers' vision beyond line of sight, and thus can know a wider range of road conditions to avoid traffic accidents at the most extent. In addition, the vehicle network can also provide services for vehicle users on entertainment and other applications. People can share interesting resources or get information about gas stations, parking lots and other related resource in a temporary virtual community.

VANETs usually consist of several key components, Roadside Unit (RSU) and On Board Unit (OBU). RSUs are deployed on both sides of the road, which not only can send group messages but also can be used as service gateway, so that each vehicle can access a remote service on the road through RSUs. And in VANETs, each vehicle is equipped with an OBU, which not only allows the vehicle to communicate with each other, namely vehicle-to-vehicle (V2V) communication, but also communicate with the roadside units (RSUs), i.e. vehicle-to-infrastructure (V2I) communication. Therefore, compared with pure traditional infrastructure-based networks, such as cellular networks, the combination of V-2-V and V-2-I communications makes VANET more promising. In the near future, it is forecasted that VANETs will be a comprehensive development platform in vehicle-centric applications [1].

Although VANETs bring us convenience, it also brings a lot of problems at the same time. In order to improve the security of communication in VANTEs, the secure key management is essential. But existing schemes [2-11] are not suitable for real

VANETs environment. DIKE [1] proposes a method to dynamically update key, which is worth learning. But its centralized network authority structure results in the problem that the scheme cannot be applied to wide VANETs. DKM [7] can avoid the authorization controversy, and reduce the size of the obsolete list. But it cannot fundamentally eliminate the impact of the obsolete list on the system efficiency, and the calculation is complex. ECCEGT-KM [8] realizes the key management without the overheads of key revocation and update. But each of its communications is dependent on DRSUs. This greatly reduces the efficiency of communication and will produce serious communication delay once in areas where DRSUs are sparse. And this scheme is unable to meet the non-repudiation. In order to compensate for the lack of existing schemes, this paper combines dynamic secret sharing and elliptic curve ElGamal threshold mechanism, to develop a new efficient and safe dynamic key management scheme in VANETs called EDKM_DSS. Compared with DKM and ECCEGT_KM, EDKM_DSS is superior in overall performance.

The remainder of this paper is organized as follows. In Section 2 we describe the new dynamic key management scheme in detail. In Section 3 we analyze the security of the proposed scheme, and study its overhead in terms of communication, computation, and storage. Finally, Section 4 gives some conclusion remarks.

2 A New Dynamic Key Management Scheme in VANETs

2.1 The Network Model

As illustrated in Fig.1, there are four kinds of members in VANETs.

(1) TA (Trust Authority)

(2) cRSU (Road Side Unit for Calculations)

(3) sRSU (Road Side Unit for Storage)

(4) OBU (On Board Unit)

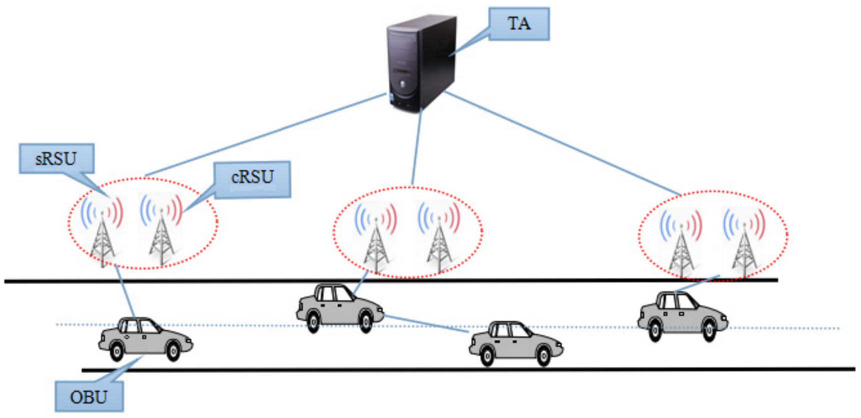


Fig. 1. The network model

2.2 The Design of EDKM_DSS Protocol

System Initialization

(1) *TA Initialization*

1) TA selects a big prime numberp_t, an elliptic curve$E_t(a_t, b_t)$ on the finite field GF(p_t) , and a random generator g_t to form the public parameter of tem(p_t, E_t, g_t), TA selects random number $\gamma \in Z_{pt}^*$as the private key for communication, and calculates the public key of systemw $= \gamma g_t$.

2) TA selects two random numbersx $\in Z_{pt}^*$, $pri_key \in Z_{pt}^*$for cRSU, sRSU and vehicle users in VANETs respectively to generate key(x, pri_key, pub_key). $x$is the identity of users, pri_key is the private key of users, $pub\_key = pri\_key \cdot g_t$ is the public key of users. The vehicle users prestore all the public key of cRSUs.

(2)*RSU Initialization*

1) The cRSU selects a big prime number p_r, an elliptic curve$E_r(a_r, b_r)$in the finite field GF(p_r) , and a random generatorsg_r, the order of g_r is q.

2) The cRSU calculates $y = mg_r$, here g_r is a generator inE_r, m is a random number selected from Z_q and kept secret. cRSU selects random numbers a_0, a_1 from Z_q, $a_0 = m$, to construct the shamir polynomial $f(x) = (a_0 + a_1 x) mod\ q$.

3) The sRSU sends x_s to *cRSU* through the secure channel, cRSU calculates the private key $(x_s, A_s)$of sRSU on the basis of x_s.The A_s is calculated according to the equation $A_s = f(x_s)\ mod\ q$, and then cRSU returns it to sRSU through the secure channel.

4) The cRSU broadcasts the public parameter $PUK = (E_r, g_r, y)$ in its coverage areas.

Encryption and Decryption

To prevent information from being modified or peeped by malicious users in the communication process, the message must be encrypted before being transmitted. The encryption process is shown as follows.

(1) In the encryption process, OBUa first calculates $b_{a-s} = (-x_a)/(x_a - x_s)$ at the basis of its own identity x_a and the identity $x_s$of sRSU.

(2) Assuming the message that will be transmitted by OBUa is M, OBUa selects a random number$r \in Z_{pt}^*$, and encrypts Musing its own private key (x_a, A_a), then gets the cipher $Enc_{(x_a, A_a)}(M) = x_s||(1 - b_{a-s})rg_r||rA_a g_r b_{a-s}||ry + M$. After that, the message $T_e||Enc_{(x_a, A_a)}(M)$ will be got by linking $Enc_{(x_a, A_a)}(M)$ with the current time-stampT_e. The signed message $Sig = x_s H\left(T_e||Enc_{(x_a, A_a)}(M)\right)$ will be got using x_s and the hash value. Finally, the message $T_e||Enc_{(x_a, A_a)}(M)||Sig$ is sent to all the users.

(3) After the users in the group receive the message $T_e||Enc_{(x_a, A_a)}(M)||Sig$, they use x_s to verify the signature and time-stamp, if $x_s H\left(T_e||Enc_{(x_a, A_a)}(M)\right) = Sig$ and $T' - T_e < \Delta T$, they will decrypt the message using the private key $(x_s, A_s)$of sRSU.

(4) In the decryption process, users first calculate $ry = A_s(1 - b_{a-s})rg_r + rA_a g_r b_{a-s}$, and then subtract ry using $ry + M$ to get the original message M.

Key Update

Key update is an important measure to ensure the vehicle users′ privacy. The following steps are required to realize key update.

(1) First, sRSU selects a random number$r_s \in Z_q$, and encrypts$M$ including key update factor$\gamma_a$with its own private key(x_s, A_s), to get the cipher $Enc_{(x_s,A_s)}(\gamma_a) = (1 - b_{s-i})r_s g_r || r_s A_s g_r b_{s-i} || r_s y + M$, here $b_{s-i} = (-x_s)/(x_s - x_i)$. And link the cipher with time-stamp T_s to get the update command message $Com\_update = T_s || Enc_{(x_s,A_s)}(\gamma_a)$. sRSU uses its own identity x_s to sign the Com_update , and gets signed message $Sig\_update = x_s H(Com\_update)$, then sends the $Com\_update || Sig\_update$ to the existing users in the group session.

(2) The users in the group session first use x_s to verify the signature and time-stamp after receiving $Com\_update || Sig\_update$, i$x_s H(Com_{update}) = Sig_update$ f and $T' - T_e < \Delta T$, they decrypt the message using their own session private key (x_i, A_i), the specific process is as follows.

First $\quad r_s A_s g_r b_{s-i} + (1 - b_{s-i}) r_s g_r \cdot A_i = r_s y$

Then $\quad (r_s y + M) - r_s = M$

At last users will get M and then extract the key update factorγ_a from M to update their own private key$A_i = A_i + \gamma_a \ mod\ q$.

Vehicle Users Join

(1) OBUa encrypts the identity x_a that got from TA with cRSU′s public key to generate the join request $Req\_join = T_a || Enc_{c.pub\_key}(x_a)$. To prevent message being tampered, OBUa signs the message with x_a and gets the result$Sig_req_join = x_a H(Req_join)$, then send $Req\_join || Sig\_req\_join$ to cRSU.

(2) After cRSU receives the request of OBUa, it decrypts the request with private key and get x_a . Then cRSU verifies the signature and time-stamp, if $x_a H(Req\_join) = Sig\_req\_join$ and $T' - T_a < \Delta T$, it traverses the revocation users list, if x_a is not in the revocation list, cRSU verifies the effectiveness of x_a. If x_a is effective, put x_a into$f(x) = (a_0 + a_1 x) mod\ q$, and got $f(x_a)$. After that, cRSU chooses a random number $\gamma_a \in Z_q$ as the key update factor, and calculates the session key A_a of OBUa according $A_a = (f(x_a) + \gamma_a) mod\ q$.

(3) cRSU encrypts (x_a, A_a, x_s, A_s) with the public key of OBUa to get $Enc_{a.pub\_key}(x_a, A_a, x_s, A_s)$, and generates the reply for joining $Rep\_join = T_c || Enc_{a.pub\_key}(x_a, A_a, x_s, A_s)$. Then cRSU signs the reply with private key $c.pri\_key$ to get the message $Sig\_rep\_join = Enc_{c.pri\_key}(H(Rep\_join))$, and sends $Rep\_join || Sig\_rep\_join$ to OBUa .

(4) At the same time, cRSU sends the key update factor γ_a and x_a to sRSU through secure channel. After receiving γ_a and x_a, sRSU first calculates b_{s-a}, then inserts (x_a, b_{s-a})into the users list UL and updates the key.

(5) After OBUa receives $Rep\_join || Sig\_rep\_join$, it first verifies the signature and time-stamp, if $H(Rep\_join) = Dec_{c.pub\_key}(Sig\_rep\_join)$ and $T' - T_c < \Delta T$, OBUa decryptsRep_join with its private key $a.pri\_key$, and stores(x_a, A_a, x_s, A_s).

Vehicle Users Leave

(1) When OUBb wants to leave from the group session, it should first encrypt its identity with the public key of sRSU $s.pub\_key$ to get the message $Req\_leave = T_b||Enc_{s.pub\_key}(x_b, "leaving")$, and signs Req_leave with its own identity to get $Sig\_req\_leave = x_b H(Req\_leave)$.Then OUBb sends $Req\_leave||Sig\_req\_leave$ to sRSU.

(2) After sRSU receives the request for leaving, it decrypts the request with its own private key to get the identity of user b. Then it verifies the signature and timestamp, if $x_b H(Req\_leave) = Sig\_req\_leave$ and $T' - T_b < \Delta T$, it finds (x_b, b_{s-b}) from UL according to x_b and delete it from UL, and updates the key for the group session.

When a user sends false information or performs other illegal activities, it is necessary to mark the user identity invalid. This time TA will send the invalid identity x_d to all the cRSU and sRSU, cRSU and sRSU add it to their revocation list, then sRSU checks UL to check if it contains (x_d, b_{s-d}), if it includes the entry, it will delete it from the user list, and then update the key for group session for all users except x_d. The communication process is shown in Fig.2.

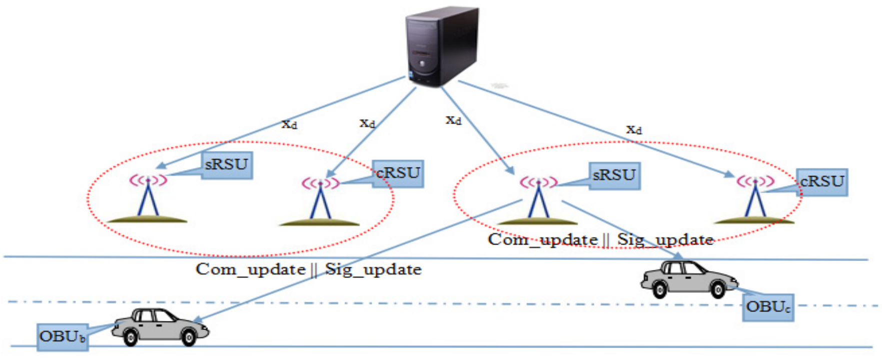


Fig. 2. Invalidate users

3 The Analysis of the EDKM-DSS Protocol

(1) Communication Overhead

In the paper, the communication overhead is measured according to the number of times of interaction. We will mainly consider the communication overhead generated between vehicle users, the key update overhead between vehicle users and RSUs. Although the communication and numeracy skills in VANETs are very capable, with the increase in the number of interaction, communication overhead will be still linearly increased. Coupled with the nodes in VANETs have been in a fast-moving process, the excessive interactions will lead to large transmission delay, so reducing the number of interactions is essential for reducing communication overhead.

Fig.3 shows the number of interactions required for users to complete one time of communication for each scenario.

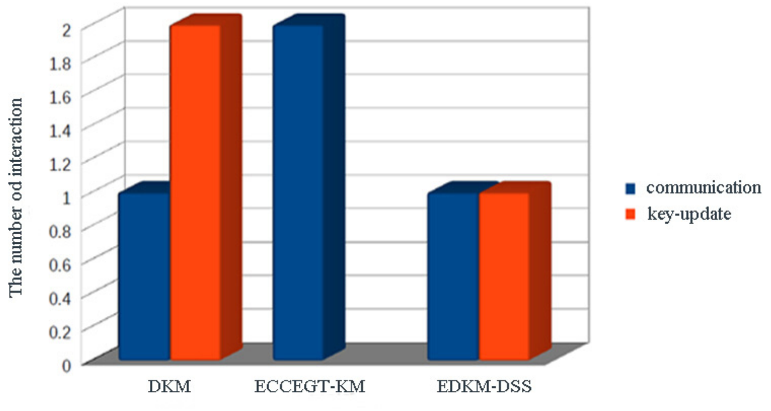


Fig. 3. Communication overhead

For DKM, in one communication process of vehicle users, the number of exchange times is 1, and in one key updating process, the number of exchange times is 2. For ECCEGT-KM, without doing any key updating, the number of exchange times in vehicle users communication is 2. For EDKM-DSS, the number of exchange times in communication and key updating process is 1 respectively. Because in the real system, the frequency of occurrence of the communication process between users is much higher than the frequency of occurrence of key update, so, on the whole, the communication overhead of EDKM-DSS program has advantages compared with the DKM and ECCEGT-KM.

(2) Computational Overhead

The numeracy skill of network node in VANETs does not need to be worried about. However VANETs is a huge system, during operations, it will need a lot of computation. Simplify of calculation will also play a vital role for the improvement of the efficiency of the protocol. To measure the computational overhead, we design and simulate the algorithm of the DKM, ECCEGT-KM and EDKM-DSS, as shown in Fig.4.

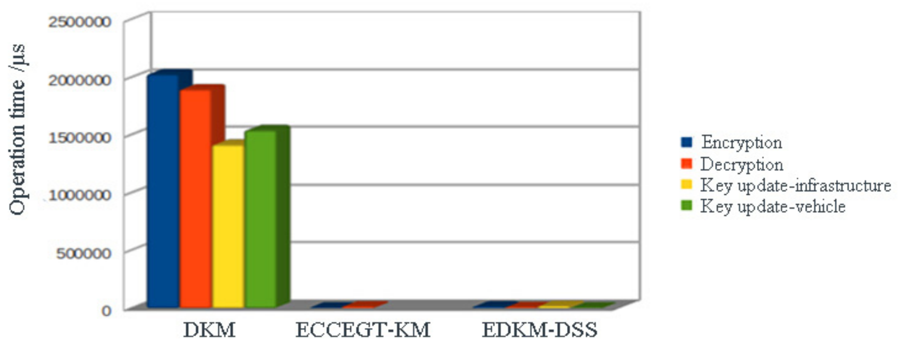


Fig. 4. Computational overhead

It can be seen from the figures that the overhead of computations in DKM is much larger than the costs of ECCEGT-KM and EDKM-DSS. The reason for the worse overhead is that, DKM applies the group signature technique with complicated computation, when signing and doing authentication.

(3) Storage Overhead

Storage overhead here mainly refers to the storage overhead of the revocation list. In DKM protocol, although the use of distributed network architecture reduces the size of the revocation list, it still inevitably uses revocation list. With the increase in the number of the user's vehicle, the size of the revocation list will rely on linear growth. Therefore the overhead spending on matching the revocation list cannot be avoided. The new proposed EDKM-DSS program and ECCEGT-KM program do not require the user to store the revocation list, so from the storage overhead perspective, EDKM-DSS program and ECCEGT-KM program are almost the same, better than DKM as shown in Fig.5.

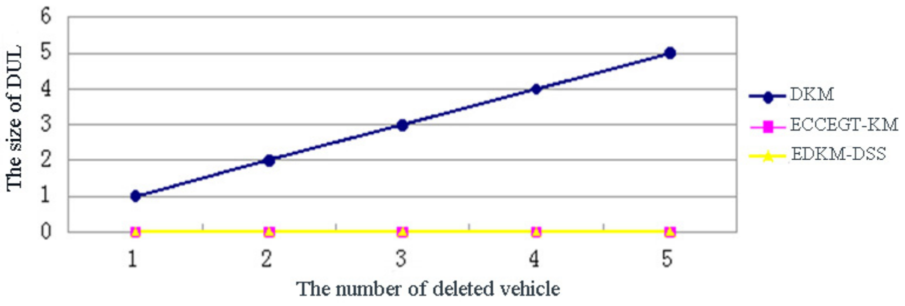


Fig. 5. Storage overheard comparison

(4) Security

The proposed EDKM_DSS program fully considers the needs of safety in VANETs, and integrates the advantages of a large number of existing programs. Compared with DKM and ECCEGT_KM, safety performance has been improved. Table 1 describes the performance comparison of the three schemes in detail.

Table 1. Performance comparison security program

Security needs	DKM	ECCEGT_KM	EDKM_DSS
Forward secrecy	No	Yes	Yes
Backward secrecy	No	Yes	Yes
Anti-eavesdrop	No	Yes	Yes
Data integrity	Yes	No	Yes
Privacy	Yes	Yes	Yes
Conditional-Anonymity	Yes	No	Yes
Verifiability	Yes	Yes	Yes
Repudiation	Yes	No	Yes

Overall, EDKM-DSS has a great advantage compared with DKM and ECCEGT-KM. Compared with DKM, EDKM-DSS greatly improves the computational overhead and storage overhead, under the premise that their overall communication overhead is basically the same. Compared with ECCEGT-KM, in the case that their computational overhead and storage overhead are almost the same, EDKM-DSS significantly reduces the communication overhead. In terms of safety performance, EDKM-DSS program fully combines with the advantages of DKM and ECCEGT_KM to further improve the security of the key management process in VANETs.

4 Conclusion

This paper studied the status of key management scheme in VANETs and found that existing schemes often considered just one aspect. They are either focus on efficiency or focus on safety, almost no proceed from the overall, and still lack of practicality. In order to compensate for the lack of existing programs, this paper combined dynamic secret sharing mechanism with elliptic curve ElGamal threshold mechanism to present a new distributed key management scheme called EDKM-DSS, and described the program in detail.

Acknowledgment. This work is supported by National Natural Science Foundation of China under Grants No.61202494, 61103203 and 61202495.

References

1. Lu, R.X., Lin, X.D., Liang, X.H., Shen, X.M.: A Dynamic Privacy-Preserving Key Management Scheme for Location-Based Services in VANETs. IEEE Transactions on Intelligent Transportation Systems 13, 127–139 (2012)
2. Raya, M., Huhaux, J.P.: Securing Vehicular Ad Hoc Networks. Journal of Computer Security 15, 39–68 (2007)
3. Lu, R.X., Lin, X.D., Zhu, H.J., Ho, P.H., Shen, X.M.: ECPP: Efficient Conditional Privacy Preservation Protocol for Secure Vehicular Communications. In: 27th IEEE International Conference on Computer Communications (INFOCOM), pp. 1229–1237. IEEE Press, USA (2008)
4. Wasef, A., Jiang, Y.X., Shen, X.M.: ECMV: Efficient Certificate Management Scheme for Vehicular Networks. In: Proceedings of the Global Communications Conference (GLOBECOM), pp. 639–643. IEEE Press, USA (2008)
5. Sun, Y.P., Lu, R.X., Lin, X.D., Shen, X.M., Su, J.S.: A Secure and Efficient Revocation Scheme for Anonymous Vehicular Communications. In: IEEE International Conference on Communications (ICC), pp. 1–6. IEEE Press, South Africa (2010)
6. Hao, Y., Cheng, Y., Ren, K.: Distributed Key Management with Protection Against RSU Compromise in Group Signature Based VANETs. In: Proceedings of the Global Communications Conference (GLOBECOM), pp. 4951–4955. IEEE Press, USA (2008)

7. Sun, Y.P., Hu, Q.L., Su, J.S.: A Distributed Key Management Scheme for the Group Signature Based on Authentication in VANETs. Computer Engineering & Science 34, 6–11 (2012)
8. Na, R., Nishide, T., Hori, Y.: Elliptic curve ElGamal Threshold-based Key Management Scheme against Compromise of Distributed RSUs for VANETs. Journal of Information Processing 20, 846–853 (2012)
9. Zhang, Y.C., Liu, W., Lou, W.J., Fang, Y.G.: Securing mobile ad hoc networks with certificateless public keys. IEEE Transactions on Dependable and Secure Computing 3, 386–399 (2006)
10. Guo, J., Baugh, J.P., Wang, S.: A Group Signature Based Secure and Privacy-Preserving Vehicular Communication Framework. In: Proceeding of the Mobile Networking for Vehicular Environments Workshop (MOVE), pp. 103–108. IEEE Press, USA (2007)
11. Lin, X.D., Sun, X.T., Ho, P.H., Shen, X.M.: GSIS: A Secure and Privacy Preserving Protocol for Vehicular Communications. IEEE Transactions on Vehicular Technology 56, 3442–3456 (2007)

Design and Implementation of Network Hard Disk

Hong Song[1], Jialong Xu[2], and Xiaoqiang Cai[1]

[1] School of Information Science and Engineering, Central South University, China
[2] School of Software, Central South University, China
songhong@csu.edu.cn, qxj11010@gmail.com, 104943476@qq.com

Abstract. In order to let the users share data freely in different places, this paper designs and realizes a new network disk system, which is called MyNDS. MyNDS uses C/S mode to construct its structure. And Winsock2 is used for clients communicating with server. When a user accesses the data storing in MyNDS server, MyNDS Driver module catches and interprets the request into a sharing file. After retrieving and communicating to the Server, MyNDS parses the request and return the correct executing result to the user. Also MyNDS uses MD5 algorithm and RSA algorithm to enhance system security.

Keywords: network storage, IRP, IOCP mechanism, memory-mapping file.

1 Introduction

Network storage systems are widely used to solve the problem of accessing data remotely. Most of the portal sites provide the service of data storage so people can share information in different places or coordinate each other in a large project through upload and download files. These storage systems enhance the working efficiency. Microsoft even sets up SkyDrive to provide a series of cloud storage service in Win8 [1-3].

For ordinary Internet users, upload and download their own data are their most usual works. The network hard disk can solve this problem well by file sharing and information transmission. So as most of people use Windows System at present, implementing a network storage system for Windows is practically and significantly.

A lightly network storage system, which is called MyNDS, is implemented to upload or download data in this paper. Through the driver programs of hard disk filter layer, MyNDS intercepts and rewrites users' IRP requests of read/write operations. And then MyNDS sent this IRP request as a socket packet to the server by inter process communication and Windows synchronization mechanism. After receiving the request packet, the server analyzes the content and executes the read/write operations. The result can then be back to the client.

The remainder of this paper is structured as follows. In section 2, we introduce the design of MyNDS, including some modules for client and server. Some key technologies used in MyNDS are described in section 3. These technologies also affect the performance of MyNDS. And Conclusion is summarized in section 4.

2 Design of MyNDS

2.1 The Overall Architecture

The MyNDS system architecture is composed of client and server, they communicate through Winsock2. The whole architecture is shown in Figure 1.

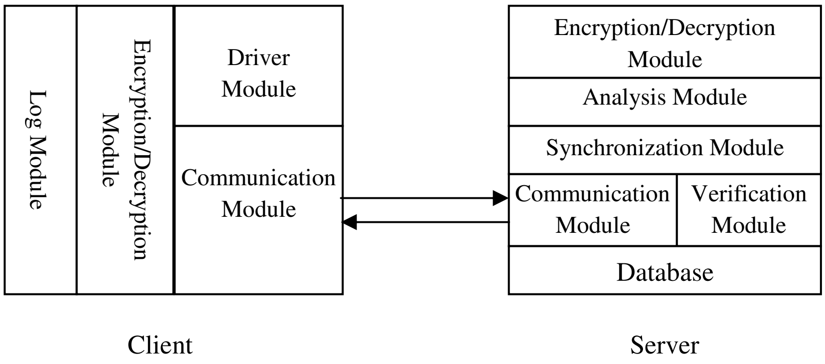


Fig. 1. The whole architecture of MyNDS

As shown in Fig. 1, the functions of the client mainly include the driving module, communication module, encryption/decryption module and log module. The server is the provider of network storage system. The functions include the driver module, communication module, verification module, synchronization module, analysis module, encryption/decryption module and database.

2.2 Design of Modules

Driver Module. The client needs to transmit user's actions in OS to the server synchronously. MyNDS uses memory-mapping file to finish this function. First, driver module of client intercepts user's operation request and stores in the memory-mapping file. Then communication module gets the request from memory-mapping file and sends to the server as a synchronous mode.

So we use a layered driver to implement the driver module. A High FiDO driver is mounted to E drive and the operations on drive E can be captured by MyNDS driver module. The module simply judges whether the operation/IRP is read or write and then copy this content to the memory-mapping file as a specific method, which is defined by MyNDS. The detail workflow is shown in Figure 2.

As shown in Fig. 2, after getting the IRP from client OS, driver module calls *MyNDS_ReadIrpDispatch*() function or *MyNDS_WriteIrpDispatch*() function to deal with the IRP request and serializes IRP requests using *StartIO*() routine. Then driver module calls *MyNDSDriverStartIO*() function to complete logical-device-creating work, memory-mapping-file-initiating work and layer-driver-initiating work. After that, driver module records the IRP information into the designated memory-mapping file. The format for IRP information is shown as Table 1.

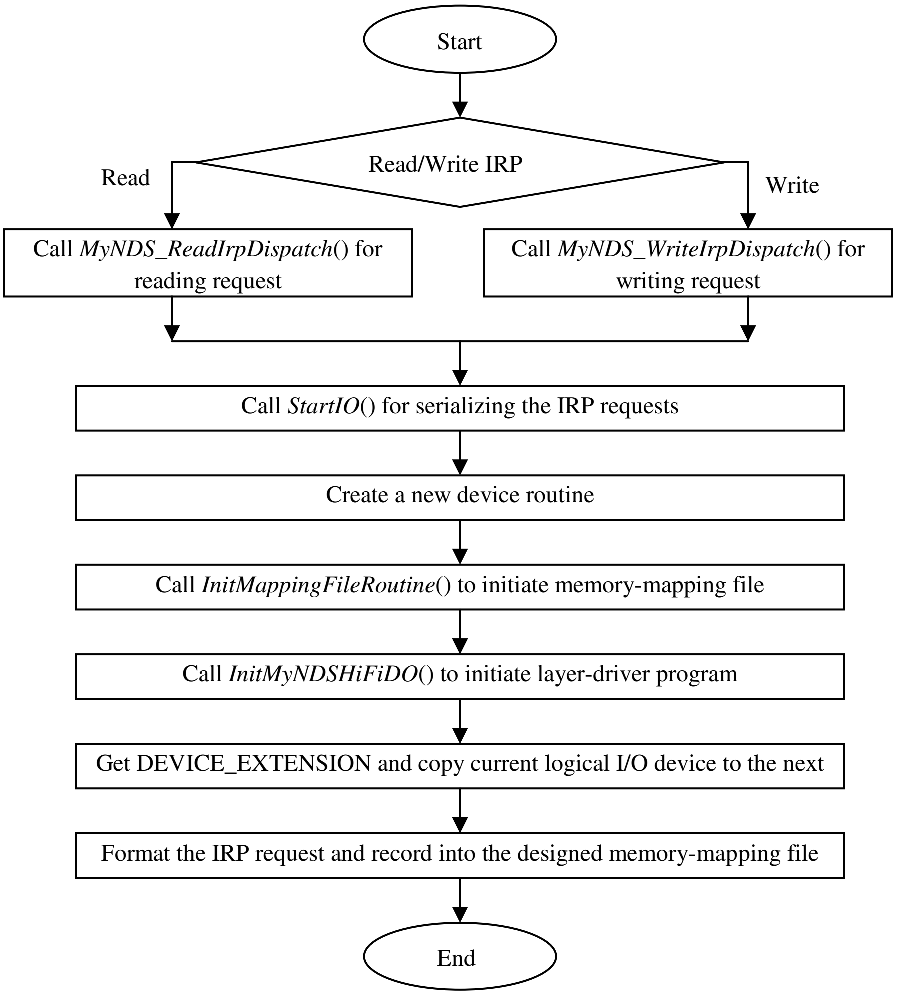

Start
Read/Write IRP
Read
Write
Call *MyNDS_ReadIrpDispatch*() for reading request
Call *MyNDS_WriteIrpDispatch*() for writing request
Call *StartIO*() for serializing the IRP requests
Create a new device routine
Call *InitMappingFileRoutine*() to initiate memory-mapping file
Call *InitMyNDSHiFiDO*() to initiate layer-driver program
Get DEVICE_EXTENSION and copy current logical I/O device to the next
Format the IRP request and record into the designed memory-mapping file
End

Fig. 2. Detail workflow of the driver module

Table 1. The format of IRP information

Segment name	Length	Function
AssociatedIrp.SystemBuff	4 Bytes	Store start addresses of *CreateFile*() or *WriteFile*()
Parameters.Write.Length	2 Bytes	Record the total bytes that needing to write
Parameters.Read.Length	2 Bytes	Record the total bytes that needing to read
IOStatus.Information	2 Bytes	Record the total bytes that read/write in the actual device

Communication Module of Client. The communication module is mainly responsible for the client connecting to the server and transmitting the mapping file content. In MyNDS, it is the bridge of data transmission for client and server. When client wants to upload information to the server, communication module gets the data from mapping file and packs to send to the server with WinSock2. And if Client wants to download, communication module stores the information to the mapping file after receiving and resolving the packages from the server. The communication module is also responsible for the connection/disconnection to server. It includes four parts, such as *InitSock()*, *SendMappingFileThread()*,*RecvMappingFileThread()* and *ExitSock()*.

The work process of *InitSock()* is shown in Fig. 3.

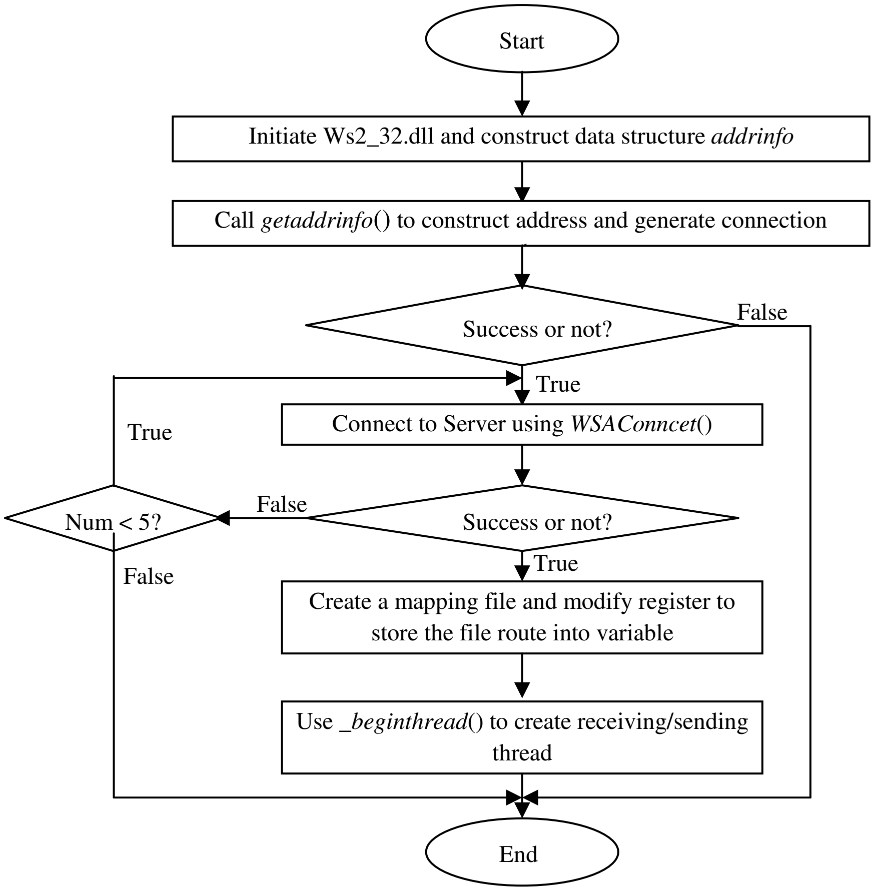

Start
Initiate Ws2_32.dll and construct data structure *addrinfo*
Call *getaddrinfo*() to construct address and generate connection
Success or not?
False
True
Connect to Server using *WSAConncet*()
True
Num < 5?
False
Success or not?
False
True
Create a mapping file and modify register to store the file route into variable
Use *_beginthread*() to create receiving/sending thread
End

Fig. 3. The process of *InitSock()*

As shown in Fig. 3, the communication module initiates Ws2_32.dll firstly and constructs the address. After that, it call socket to generate connecting socket and try to connect to server using *WSAConnect*(). A sharing file will be created for succeeding, which includes the file name and file path.

SendMappingFileThread() and *RecvMappingFileThread*() are the kernel functions. The two functions finish actions of uploading/downloading. Because of multithread environment, the functions use three events to synchronize. The three events are *kevtMappingFileSC*(), *kevtMappingFileCS*() and *kevtConent*(). When *kevMappingFileSC*() and *kevContent*() are in notification status but *kevtMappingFileCS*() is absent, the communication module can send data. If *kevMappingFileSC*() is in notification status but *kevtMappingFileCS*() and *kevContent*() are absent, the communication module can receive data from server.

ExitSock() is used to recycle the resources, such as sharing file, thread resource and etc. . If communication module wants to exit, it sends a shutdown signal to server firstly and then it will close the socket and file handle. At last, it terminates the sending thread and receiving thread.

Overall Structure of Server. The server is the core of MyNDS system. It is mainly consist by the log module, analysis module, data module, encryption/decryption module, communication module, authentication module, synchronization module. Among them, the log module and an encryption/decryption module are running after MyNDS is started. And the other modules are interacted with each other. The relationship of each module is shown in Fig. 4.

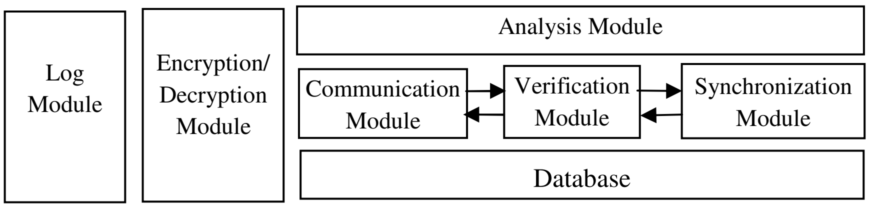


Fig. 4. The relationship of each module

Log module will monitor and record the operations of MyNDS. Every record is composed to a specified string format and will be written to the log file. In addition, it is also used in the event of a system crash or other problems. Three types of log are USERMSG, ERROR and RUNTIMEINFO. The log module in this system is mainly to manipulate the string, including the string split, combination, and replication, formal.

In order to ensure the security of communication, we design Encryption/Decryption module. MD5 algorithm and RSA algorithm are used to implement this function. MD5 algorithm is used to encryption the data stored into database, while RSA algorithm is for data transmission on network. It can avoid interception or forged attack.

Analysis module translates the encryption data to Read/Write operations, which are IRP requests from client. After checking parameters of encryption package, analysis module will get the fixed-length character string from designed start address using *memcopy*() and will add "\0" at the end of the string.

Communication module of server is to correspond with the client. IOCP mechanism and overlay I/O model are used to improve the performance and efficiency of server.

Verification module aims at affirming not only the correct user and password, but also the correct data package header, the correct user's instructions, the correct source and destination, as well as the correct authorization. The module will need the information from database.

Synchronization module is used to synchronize the operations both of client and server. It will uses *ProcessRWMsg*() to finish the work. The workflow of *ProcessRWMsg*() is shown in Fig. 5.

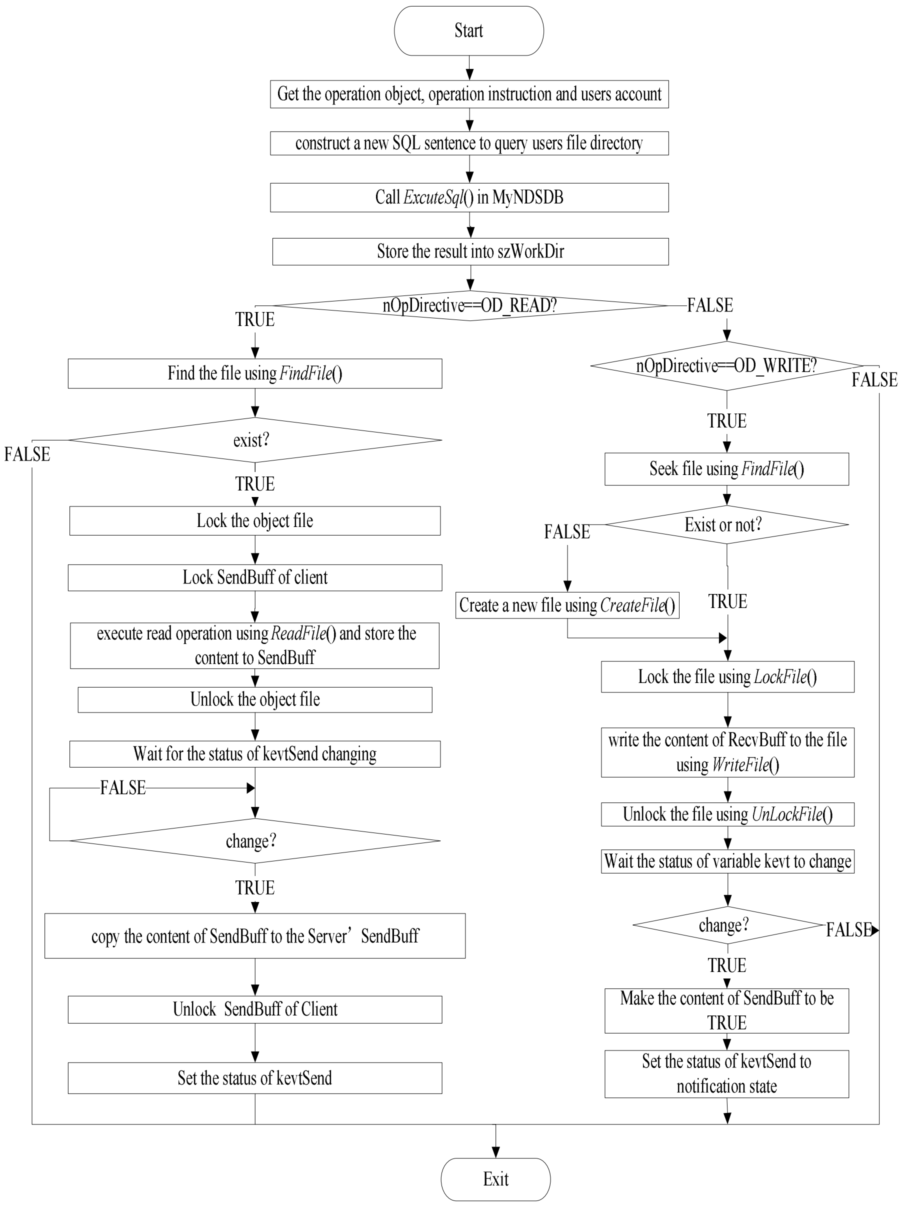

Start
Get the operation object, operation instruction and users account
construct a new SQL sentence to query users file directory
Call ExcuteSql() in MyNDSDB
Store the result into szWorkDir
nOpDirective==OD_READ?
TRUE
FALSE
Find the file using FindFile()
exist?
FALSE
TRUE
Lock the object file
Lock SendBuff of client
execute read operation using ReadFile() and store the content to SendBuff
Unlock the object file
Wait for the status of kevtSend changing
FALSE
change?
TRUE
copy the content of SendBuff to the Server' SendBuff
Unlock SendBuff of Client
Set the status of kevtSend
nOpDirective==OD_WRITE?
FALSE
TRUE
Seek file using FindFile()
Exist or not?
FALSE
Create a new file using CreateFile()
TRUE
Lock the file using LockFile()
write the content of RecvBuff to the file using WriteFile()
Unlock the file using UnLockFile()
Wait the status of variable kevt to change
change?
FALSE
TRUE
Make the content of SendBuff to be TRUE
Set the status of kevtSend to notification state
Exit

Fig. 5. Relationship between synchronization module designs

Database Design. Database is to store user information and log data query information. So, there are some tables in the entire MyNDS database. Two more important table structures are described in Table 2 and Table 3.

Table 2. User information table

Field name	Field type	Filed length	Allow null values
ID	Int	11	FALSE
Account	Varchar	20	FALSE
Pwd	Varchar	50	FALSE
Dir	Varchar	512	FALSE
RealName	Varchar	20	TRUE
CardID	Char	18	TRUE
MailAddr	Varchar	128	TRUE
PhoneNum	Char	11	TRUE
LastLoginTime	Date	4	FALSE

Table 3. Log table

Field name	Field type	Field length	Allow null values
ID	int	11	FALSE
LogTime	varchar	4	FALSE
Dir	varchar	512	FALSE

In database, we also design some functions to help MyNDS finish establishing connection with database, querying, extracting the result and inserting the data into database.

3 The Key Technology to Realize MyNDS

3.1 Transfer Parallel IRP Requests to Serial IRP Requests

In driver programming, IRP processing is divided into serial and parallel processing. As for serial port equipment, serial IRP processing is required for not causing confusion. However, the Windows operating system is preemptive and multitask system. The user's operation is parallel processing. So converting the parallel IRP to serial IRP is needed. We use the *StartIO*() routine in Windows programming to complete the conversion.

The conversion process of *StartIO*() routine can be described as follows. First, IRP requests are arranged to a FCFS queue. Then driver module selects the first IRP request from the queue. For the queue, the parallel IRP requests can be changed to serial IRP requests.

We design a new function named *MyNDSDriverStartIO*() to finish this work. Firstly, *MyNDSDriverStartIO*() gets a spin lock and judge whether there is an IRP request. If there is an IRP request, *MyNDSDriverStartIO*() then confirms whether the request is read request or write request. *WriteRoutine*() or *ReadRoutine*() will be called separately to deal with different request. The detail working process is described in Fig. 6.

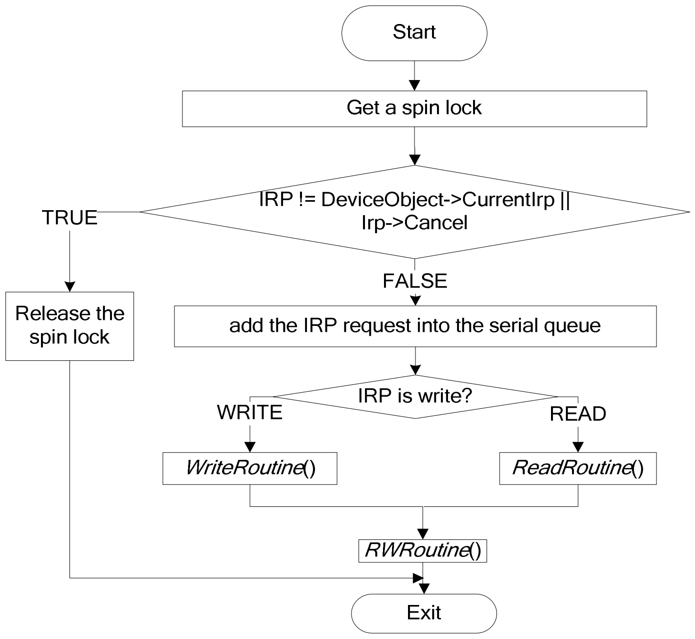


Fig. 6. Working process of *MyNDSDriverStartIO*()

3.2 Using IOCP Mechanism

IOCP[4], which is I/O Completion Port, is used to process large amount of communication requests for better QoS. The core idea of IOCP is delivering all requests into a message queue and using some worker threads to process the requests in parallel. Only a few of threads must handle a large number of I/O requests and the CPU time is not waste. And IOCP can improve the resource utilization rate.

Implementation steps of IOCP mechanism is described as follow:

- Create a completion port object in the main thread.
- The completion port object is bind to the thread.
- Use the *WSAAccept*() function to create a overlapped socket, and then bind it to a completion port object..
- Send and receive data through the *WSASend*() and *WSARecv*() functions.

In MyNDS, we use BuildIOCP to implement the IOCP mechanism. We firstly call *CreateIoCompletionPort*() to create a IOCP port, which is named MyNDSIOCP. Then MyNDS creates worker threads twice as much as the number of CPU and binds the worker threads with MyNDSIOCP by *GetQueuedCompletionPort*(). After accepting request by *WSAAccept*() and getting an *AcceptSocket*() for communication, *CreateIoCompletionPort*() function will bind the *AcceptSocket*() with MyNDSIOCP. So the worker thread can call *GetQueuedCompletionPort*() to process I/O package retrieved from IOCP. The construction process of BuildIOCP is shown as Fig. 7.

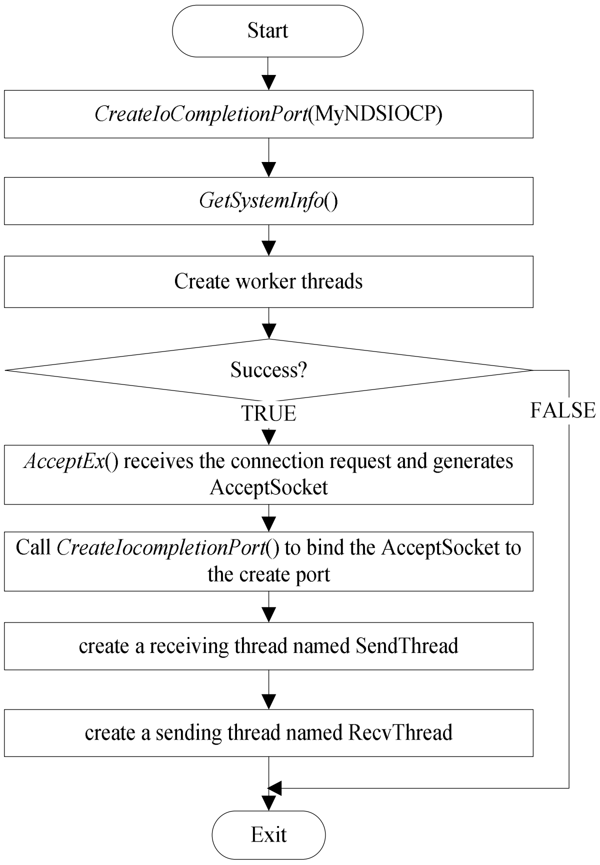

Start
CreateIoCompletionPort(MyNDSIOCP)
GetSystemInfo()
Create worker threads
Success?
TRUE
FALSE
AcceptEx() receives the connection request and generates AcceptSocket
Call CreateIocompletionPort() to bind the AcceptSocket to the create port
create a receiving thread named SendThread
create a sending thread named RecvThread
Exit

Fig. 7. Working process of BuildIOCP

3.3 Synchronization in MyNDS

Synchronization is needed in MyNDS because many operations in MyNDS is not reentrant and the processing sequence of these operations will affect the result directly, such as the queue operations in *StartIO*() routine.

In MyNDS, spin lock will be used to process synchronization, just like operations on critical resources. If spin lock is running, only the current thread can use the resource while other threads can't access the resource. Different from critical resources, the rejected threads will always occupy CPU to check whether the resource is released.

Synchronous operation generally occurs in the dispatch function. A sharing spin lock will be used among different functions, and it will be transferred by the signal DEVOBJ_EXTENSION.

Initialization will be done by function *KeInitializeSpinLock*() before using the spin lock. *KeAcquireSpinLock*() function is used to operate the spin lock. Release the spin lock is realized by *KeReleaseSpinLock*() function.

3.4 Inter-Process Communication

There are some methods for inter-process communication in Windows operating system, such as Clipboard, COM, Data Copy, DDE, File Mapping, Mail slots, Pipes, RPC, Windows Sockets[5]. In MyNDS, we mainly use File Mapping to realize the inter-process communication.

File mapping can associate a file's contents with a portion of the virtual address space of a process. The process uses a simple pointer to realize checking and modification of the content of files. When two or more processes access same mapping file, each process's operations are in its address space, as well as reading/modifying file contents. In the multitask environment, the process must use synchronization objects (for example, semaphore) to avoid data conflict.

We map a particular file to a named, sharing memory space. All processes can access this memory space with the same file-mapping object. That is to say, processes open the same file mapping object, all operations of this mapping file are operate to physical memory contents of the mapping file.

The process of using the memory-mapping file can be described as follow:

- Create or open a sharing kernel file object. Usually we use the *CreateFile*() function.
- Create a new file-mapping kernel object and send the file size to the system. *CreatFileMapping*() function can finish the work.
- System map the overall or part of the file-mapping object to the address space of designed process. *MapViewOfFile*() function is used.
- *UnmapViewOfFile*() function notifies system to release the object.
- Close the file object by *CloseHandle*() function.
- Close the kernel object by *CloseHandle*() function.

4 Conclusion

Network storage is an offsite storage mode. It can become a working platform after contacting with cloud Service. As an example of network storage system, MyNDS realizes the file upload/download. It can guarantee the communication security and the security of data storage, as well as easier management and function extension.

We talk about the design and implementation of MyNDS. It is divided into Client and Server. The client intercepts user instructions in driver module and sends the instructions to server by communication module. The server, however, processes the instructions and returns the result to the client through verification module, synchronization module, and Encryption/Decryption module and so on. In order to improving the performance of MyNDS, we also using IRP requests Serialization, IOCP mechanism and memory-mapping file for inter-process communication. Future works will be focused on the security of MyNDS.

Acknowledgement. This paper is supported in part by International S&T Cooperation Program of China under Grant No. 2013DFB10070 and National Natural Science Foundation of China under Grant No.61202495.

References

1. Ma, R.: Analyse the development process of network hard disk from the local to the clouds, `http://soft.zol.com.cn/294/2941233_all.html#p2947154`
2. Zhang, Y.X., Zhou, Y.Z.: Transparent Computing: Spatio-Temporal Extension on von Neumann Architecture for Cloud Services. Tsinghua Science and Technology 18(1), 10–21 (2013)
3. Lin, F., Piao, H.J., Ling, H., Jin, B.: A network disk encryption with dynamic encryption key. In: 2010 International Conference on Computer Design and Applications (ICCDA), pp. 605–614. IEEE Press, China (2010)
4. Jiang, S.X., Wang, Y.: IOCP application in server development. Information Security and Communications Privacy 3, 72–75 (2010)
5. Microsoft MSDN- Inter-process Communications, `http://MS.MSDNQTR.v90.chs/ipc/base/interprocess_communications.htm`
6. Fu, P., Zhen, B.W.: Some Common Realizing Methods and Analysis of the Communication between the Processing Based on the Windows. Microcomputer Applications 28, 767–770 (2007)
7. Gao, Y., Zhang, Y.X., Zhou, Y.Z.: Building a virtual machine-based network storage system for transparent computing. In: Proc of 2012 International Conference on Computer Science and Service System, pp. 2342–2344. IEEE Computer Society Press, China (2012)
8. Zhang, Y.X., Zhou, Y.Z.: A New Cloud Operating System: Design and Implementation Based on Transparent Computing. Acta Electronica Sinica 39, 985–990 (2011)
9. Hou, C.M., Chen, B.: Key Techniques of Windows Kernel Rootkit Based on Layer Drivers. J. of Jishou University (Natural Science Ed.) 4, 48–51 (2009)
10. Gao, Y., Zhang, Y.X., Zhou, Y.Z.: Performance Analysis of Virtual Disk System for Transparent Computing. In: Proc of the 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, pp. 470–477. IEEE Computer Society Press, Japan (2012)
11. Tian, Y., Lin, C., Chen, Z., Wan, J.X., Peng, X.H.: Performance Evaluation and Dynamic Optimization of Speed Scaling on Web Servers in Cloud Computing 18(3), 298–307 (2013)

Combining Supervised and Unsupervised Learning for Automatic Attack Signature Generation System

Lili Yang, Jie Wang[*], and Ping Zhong

School of Information Science and Engineering, Central South University, China
{liliyang,jwang}@mail.csu.edu.cn

Abstract. Signature-based intrusion detection system is currently used widely, but it is dependent on high quality and complete attack signature database. Despite a great number of automatic attack feature extraction system has been proposed, however, with the progress of attack technology, automatic attack signature generation system research is still an open problem. This paper presents a novel combining supervised and unsupervised learning for automatic attack signature generation system based on the transport layer and the network layer statistics feature, and the system outputs the signature sets in feedback way. Finally we demonstrate the effectiveness of the model by using network data from the laboratory and Darpa2000 datasets.

Keywords: IDS, attack signature, automatic extraction, abnormal flow.

1 Introduction

In recent years, the Internet information security consciousness is gradually improved and the popularity of security software reduces the occurrence probability of the same security events, but the emerging of all kinds of malware variants or new malicious programs makes the occurrence probability of security event still high [1][2]. In such a serious information security environment, intrusion detection technology attracts more and more people's attention. Since the 1990 s, research and development of intrusion detection system presents a prosperous situation. Because of the characteristics of simple, efficient and accurate, IDS based on the signature is widely used. It depends on high quality and complete attack signature database, so the fast and accurate automatic attack signature generation system is still an important research direction in the field of network security.

The goal of automatic attack signature generation system is to find automatically new attack and extract characteristics of the new attack which can be used in IDS [3][4]. In 2003 Kreibich put forward the first automatic attack signature generation system Honeycomb [5], after that many systems have been proposed and implemented.

Since different automatic extraction technologies are used, automatic attack signature generation systems include mainly the network-based signature generation

[*] Corresponding author.

systems (NSG) and the host-based signature generation systems (HSG). The NSG system is deployed on the Internet, and it extracts signatures by analyzing the network data. Here, the signature points to a binary string that describes the attack with composition, distribution, or frequency. The typical NSG systems are as follows: Honeycomb, PAYL, Autograph, EarlyBird check-in, Pol, ygraph, Nemean, PADS, Hamsa, SRE, etc. The HSG system usually is deployed on a host computer, and detects the abnormality of the host and uses the collection of information on a host computer to extract the signature of the attack. The typical HSG systems are as follows: FLIPS, TaintCheck, Viglante, ARBOR, ADRMCV, COVERS, HACQIT, Packet vaccine etc [6].

Our goal is to find attack by detecting abnormal data flow in network, and extract the accurate attack signature from abnormal data. Therefore, we develop an automatic and based-network attack signature generation system model. Traditional automatic attack signature generation model adopt honeypot or classifier constructed by DPI technology or based on payload technology to identify abnormal traffic [7][8]. But the classifier cannot well identify abnormal traffic with encryption and variant. Meanwhile, honeypot need a long time to respond to worm outbreak and it may contain noise in the captured sample. In our model, we use decision tree algorithm, a supervised learning method, to construct a classifier based on network layer and transport layer statistic characteristics of network flow. Many Experiments [9][10] show that the classifier can identify abnormal data stream very well. After getting abnormal data flow, we use the unsupervised machine learning method to cluster abnormal data stream into more classes, and no similarities between each cluster. We will extract the set that can describe attack from each cluster. Here, we extract the bidirectional data flow character feature set. First, we need to extract the public substring which length is greater than 3, and use the subset of a certain frequency range to test the sample. When the sum of the false positive rate and the false negative rate is the lowest, we choose the frequency range of subset as the output from the sample. To sum up, the main contribution of our works are as follows:

1) We put forward a novel attack signature automatic generation system by combining supervised learning and unsupervised learning, which is different from tradition methods in abnormal flow identification.

2) We use feedback mechanism to confirm a frequency range of subset of public substring.

2 Challenges and Motivation

The current automatic attack signature generation system is poor in facing plenty of deformation or encryption attack. Due to this challenge, this paper seeks to develop an automatic attack signature generation model to overcome the problem.

3 Proposed Framewoek

Model of automatic attack signature generation system is designed based on supervised and unsupervised learning. Figure 1 shows the proposed system model. We use the decision tree classifier which is constructed by supervised learning to identify abnormal data flow. And unsupervised learning is used to cluster abnormal flow. The system contains the following modules: (i)basic packet information processing module, (ii) flow generation module, (iii) flow-level statistical feature extraction module, (iv) abnormal flow identification module, (v) abnormal flow clustering module, (vi) abnormal packet payload extraction module, (vii) character feature extraction and selection module.

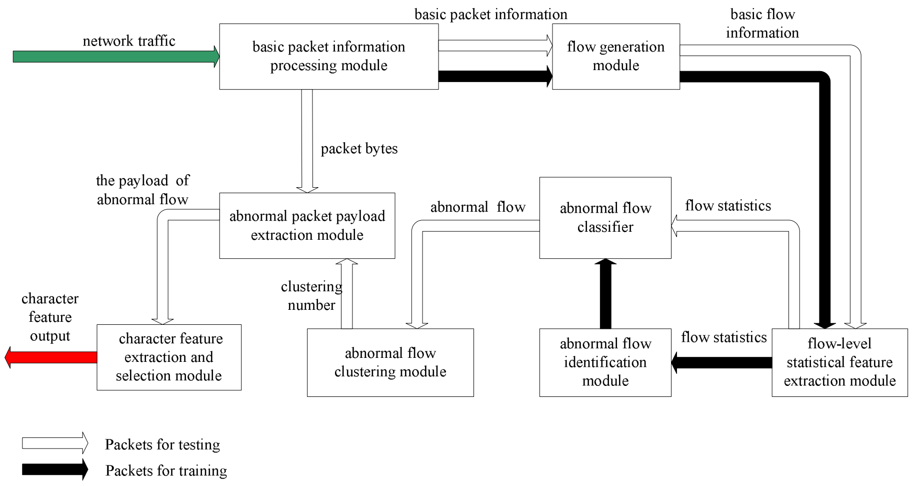

network traffic
basic packet information
basic packet information processing module
flow generation module
basic flow information
packet bytes
the payload of abnormal flow
abnormal packet payload extraction module
abnormal flow
abnormal flow classifier
flow statistics
clustering number
character feature output
character feature extraction and selection module
abnormal flow clustering module
abnormal flow identification module
flow statistics
flow-level statistical feature extraction module
Packets for testing
Packets for training

Fig. 1. Automatic attack signature generation system model

First, we need train the abnormal flow classifier. Specific process is as follows: the basic packet information processing module extracts basic information from packets for training. The flow generation module use basic packet information to generate flow, and output the basic flow information. The flow-level statistical feature extraction module obtains basic flow information to extract the statistical information and send it to abnormal flow identification module. Specially, the flow statistics come from transport layer and network layer of the TCP flow, such as the total number of packets, maximum packet size and the total number of SYN (see Table 1). Then the abnormal flow identification module trains the decision tree classifier with decision tree algorithm.

Second, we use trained abnormal flow classifier to classify packets for testing, and extract the character feature. Packets are sent to the basic information module to extract basic packet attribute and packet bytes. The flow generation module will use the basic packet information generate flow and output basic flow information. The flow-level statistical feature extraction module extracts flow statistics by using the basic flow information. At this point, the tested flow statistics information is passed to the abnormal flow classifier, rather than the abnormal flow identification module. The abnormal flow classifier will classify flow, and send the abnormal flow to abnormal flow clustering module. The abnormal flow clustering module will gather similar degree of flow and output results. Abnormal packet payload extraction module extracts the C2S or S2C sequence in cluster. Specially, C2S means the information transmission direction is client to server, S2C means the information transmission direction of server to client. In the end, character feature extraction and selection module extract character feature from information sequence. The concrete construction of model will be explained in the later.

Table 1. Flow-level Feature Generated by the Flow of Statistical Feature Extraction Module

features Name	features Description
pkts_c2s、pkts_s2c	total number of packets
pkt_noPayload_c2s、pkt_noPayload_s2c	total number of packets without payload
bytes_c2s、bytes_s2c	total number of bytes transferred
pay_bytes_c2s、pay_bytes_s2c	total number bytes from all payloads
duration_c2s、duration_s2c	flow duration
maxsz_c2s、maxsz_s2c	maximum packet size
minsz_c2s、minsz_s2c	minimum packet size
avfsz_c2s、avfsz_s2c	average packet size
stdsz_c2s、stdsz_s2c	standard deviation of packet size
IAT_c2s、IAT_s2c	average inter-arrival time
maxpy_c2s、maxpy_s2c	maximum payload size
minpy_c2s、minpy_s2c	minimum payload size
avgpy_c2s、avgpy_s2c	average payload size
stdpy_c2s、stdpy_s2c	standard deviation of payload size
synflag_c2s、synflag_s2c	total number of SYN
rstfalg_c2s、rstfalg_s2c	total number of RST
pushflag_c2s、pushflag_s2c	total number of PSH
finflag_c2s、finflag_s2c	total number of FIN

3.1 Abnormal Flow Identification Module

This module uses the transport layer and network layer flow-level statistical features to construct classifier with the supervised learning which is decision tree algorithm. The advantage of this method is to identify the encryption and deformation flow. The decision tree is selected because the decision tree classification rules of has the characteristics of high accuracy and easily understand.

3.2 Abnormal Flow Clustering Module

In this section, we use K-Means algorithm to cluster abnormal flow. The K-Means algorithm process is as follows:

If the dataset D contains n objects in Euclid space, the division method distribute the objects in D to the k clusters C_1, ..., C_k, and let $C_j \subset D$ and $C_i \cap C_j = \Phi$, where $i, j >= 1$ and $i, j <= k$. Let c_i stand for the cluster C_i, the distance between object $p \in C_i$ with center c_i is denoted by $dist(p, c_i)$, see in equation (1).

$$dist(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (1)$$

The K-Means algorithm initially chooses k objects randomly, and the selected object represents the center of cluster. For the rest of objects, according to the Euclidean distance of each cluster center, they will be assigned to the most similar cluster, the similarity is measured by $dist(p, c_i)$. Then, the algorithm iteratively changes the cluster. It uses the object which is assigned to the cluster in the last iteration to calculate a new mean for the cluster, and the new mean will be the new cluster center. When the cluster is the same as the previous, the algorithm is over. The detail of K-Means algorithm is shown in figure 2.

```
Algorithm: K- Means.
Input:
1) K: the number of clusters;
2) D: the dataset containing n objects.
Output: K clusters
Methods:
1) Select the initial cluster center from k objects;
2) Repeat
3) According to the mean of objects, let remainder object is assigned to the most similar cluster;
4) Recalculate the mean of objects in each cluster and update it;
5) Until no longer change.
```

Fig. 2. K-Means algorithm

3.3 Character Feature Extraction and Selection Module

Character feature extraction and selection module is divided into two phases. In the first phase, module extracts the direction of C2S or S2C public substring which is longer than 3 characters. In the second stage, the module detects attack sample by using the subset of public substring with a certain frequency range. When the sum of the false positives rate and the false negative rate is lowest, module outputs the range of frequency substring as the attack signature. The Detail of character feature extraction and selection algorithm is shown in figure 3.

```
Algorithm: Character feature extraction and selection module algorithm.
Input:
1) $Cluster_i$, the i-th cluster. Where $Cluster_i = \{C2S_{Cluster_i}, S2C_{Cluster_i}\}$, and $C2S_{Cluster_i}$
   denoted the information of C2S direction, $S2C_{Cluster_i}$ denoted the information of S2C direction.
2) $PublicSequence_{len \geq 3}(x, x')$, it extracts public substring which is longer than 3 characters
   between $x$ and $x'$.
Output: $Features_{Cluster_i} | (k_{\min}, k_{\max})$, where $(k_{\min}, k_{\max})$ is the frequency range of substring.
Methods:
1) If $Sequence_m \in C2S_{Cluster_i}$, $Sequence_n \in C2S_{Cluster_i}$ $m \neq n$ and then
2)     $Features \leftarrow PublicSequence_{len \geq 3}(Sequence_m, Sequence_n)$
3) Delete the repeat sequences of the $Features$;
4)Calculate the frequency of public substring in $Features$;
5) Let the substring of the range of $(k_{\min}, k_{\max})$ detects related sample. When the sum of
   the false negative rate and the false positive is lowest, the module outputs the substrings
   $Features_{Cluster_i} | (k_{\min}, k_{\max})$;
6) Repeat aforementioned process for the $S2C_{Cluster_i}$ in $Cluster_i$.
```

Fig. 3. Character feature extraction and selection module algorithm

4 Experimental Evalutation

In this section, we present the experimental results displaying the performance of the proposed model. First, we simply introduce the source of experimental data. Second, we evaluate the effectiveness of abnormal data flow classifier in dealing with indentifying abnormal network traffic. Third, we evaluate the effectiveness of abnormal data flow clustering module. In the end, we evaluate the effectiveness of the system model with character feature Extraction and selection experiment result.

4.1 Data

The proposed model is evaluated by using network traffic which is shown in table 2.

First, we adopted normal network traffic and Port_Scan in the lab. The phase-4-dump-inside and phase-5-dump-inside all come from the first attack scenarios of classical Darpa2000 intrusion detection dataset. This attack scenario includes multiple networks and audit sessions. specially, the sessions are divided into five stages: detect network, compromise hosts with Solaris sadmind, install mstream DDoS Trojan horse software, launch DDoS attack. The Phase-4-dump-inside dataset comes from the fourth stage, namely it is the phase of the installing the Trojan horse mstream DDoS software; Phrase-5-dump comes from the fifth stage which was launching DDoS attack stage.

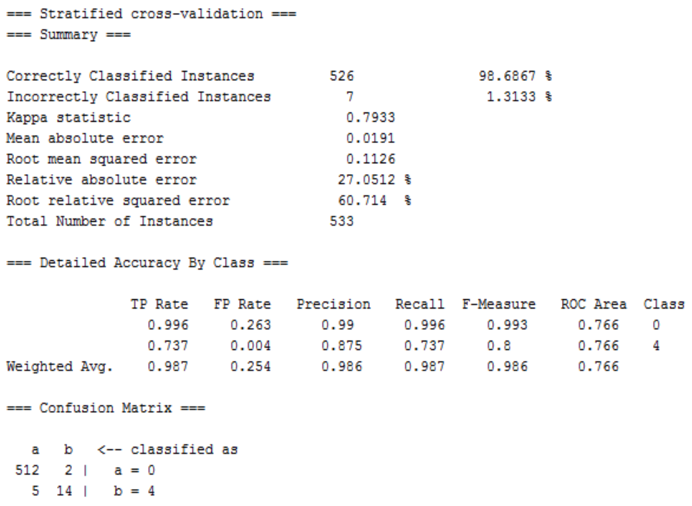
Due to the large amounts of attack traffic is based on the TCP protocol, the network traffic in the experiment is TCP flow.

Table 2. source of dataset

The dataset	Dataset description
Normal	Adopting in Laboratory
Port_Scan	NMAP collecting in Laboratory
Phase-4-dump-inside[11]	Intranet dataset of phase 4 of scenario 1 in DARPA intrusion detection
Phase-5-dump-inside[11]	Intranet dataset of phase 5 of scenario 1 in DARPA intrusion detection

4.2 Evaluating the Effectiveness of Abnormal Flow Classifier

In this section, the dataset which mixes normal with phase-4-dump-inside is sent to the abnormal flow identification module. After that we get an abnormal classifier, and we use the way of crossing validation to evaluate the effectiveness of abnormal flow classifier. See figure 3, the figure shows abnormal data flow classifier reached more than 98% of the correctly classified instance. Therefore, the way of using decision tree classifier to construct abnormal flow identification module is feasible and effective in identifying abnormal flow.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         526               98.6867 %
Incorrectly Classified Instances         7                1.3133 %
Kappa statistic                          0.7933
Mean absolute error                      0.0191
Root mean squared error                  0.1126
Relative absolute error                 27.0512 %
Root relative squared error             60.714  %
Total Number of Instances              533

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.996     0.263      0.99      0.996     0.993      0.766    0
                 0.737     0.004      0.875     0.737     0.8        0.766    4
Weighted Avg.    0.987     0.254      0.986     0.987     0.986      0.766

=== Confusion Matrix ===

   a   b   <-- classified as
 512   2 |   a = 0
   5  14 |   b = 4
```

Fig. 4. the result of verifying abnormal classifier with cross validation

4.3 Evaluating the Effectiveness of Abnormal Flow Clustering Module

We input the dataset which mixes phase-4-dump_inside with Phase-5-dump_inside into the abnormal flow clustering module in this section, and the result is shown in table 3.

Table 3. the result of clustering

category	Phase-4-dump-inside	Phase-5-dump-inside
Before clustering	19	32
After clustering	15	36
Error clustering	5	1

According to the anticipated target, the module can separate the two phase flow automatically. But the experimental result shows that the clustering effect is not very well. In table 3, the error clustering refers that after clustering the instance of phase-4-dump-inside is classified as the instance of phase-5-dump-inside, and the instance of phase-5-dump-inside is classified as the instance of phase-4-dump-inside.

We assume Right probability after clustering = right number after clustering / category number before clustering, the instance of phase-4-dump-inside dataset correctly clustering probability is 52.6%, but the instance of phase-5-dump-inside correctly clustering probability is 96%. In spite of K - Means algorithm is not very well in classifying two types of abnormal dataset. But it is good for extracting attack signature because the instance in each cluster has the similarity, and the similarity can help system extracts more accurate signature.

4.4 Character Feature Extraction and Selection Experiment Result

In this section, we respectively use phase-4-dump_inside and phase-5-dump_inside dataset to experiment. First, we do experiment by using phase-4-dump_insde dataset.

(1) The Experiment by Using Phase-4-Dump-Inside Dataset.
There are 19 flows after generating TCP flow in flow generation module. Table 4 shows the result of clustering by using phase-4-dump-inside dataset.

Table 4. the result of clustering by using phase-4-dump-inside dataset

No.	Label	Count
1	Cluster0	4
2	Cluster1	5
3	Cluster2	10

Due to Cluster0 without transport information, we experiment by using the C2S direction of cluster1 and cluster2. The result show in figure 5 and figure 6. We choose the subset of substring which is longer than 3 characters to detect the related samples.

See figure 5 and figure 6, the set of substring which is represented by the shortest cylindrical surface is as the attack signature set. Table 5 shows the attack signature of phase-4-dump-inside dataset. It displays the attacker was installing mstream Trojan software on the host by using remote desktop. Since the phase-4-dump-inside is the phase which was installing mstream software, the extracted set can describe it.

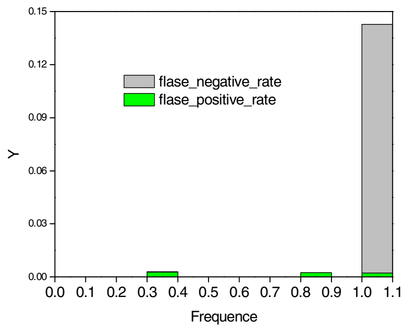


Fig. 5. The result of feature extraction experiment by using cluster1 from Phase-4-dump_inside dataset

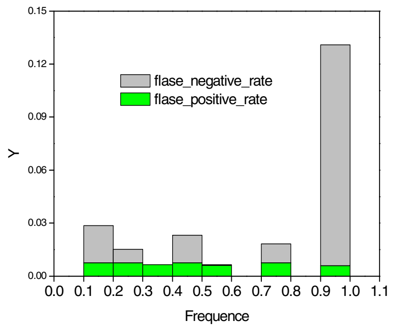


Fig. 6. The result of feature extraction experiment by using cluster2 from Phase-4-dump_inside dataset

Table 5. the attack signature of phase-4-dump-inside dataset

sample	frequency range	attack signature
cluster1	[0.3,0.4)	✓ Uroot,root,rcp -f /.sim/home/jhaines/ATTACKS/mstream/solaris/ ✓ er-sol …
cluster2	[0.5,0.6)	✓ rcp ✓ /.sim/home/jhaines/ATTACKS/mstream/solaris/ …

(2) The Experiment by Using Phase-5-Dump-Inside Dataset.
In this section, we generate TCP flow by using packets from phase-5-dump-inside dataset, then put it into the abnormal flow identification module and abnormal flow clustering module. The table 6 shows the clustering result.

Table 6. the result of clustering by using phase-5-dump-inside dataset

No.	Label	Count
1	Cluster0	4
2	Cluster1	15
3	Cluster2	13

As shown in the figure 7, figure 8 and figure 9, these figures respectively display the result of feature extraction experiment. We detect samples with a frequency range of substring which is longer than 3 characters, and when the sum of the false positive rate and the false negative rate is lowest, we will choose substrings represented by the shortest cylindrical surface as the attack signature, just like the experiment of phase-4-dump_inside dataset.

The table 7 lists some attack signature. From this table we can find the attacker is visiting a web site. Because phase-5-dump-inside dataset belongs the phase of launching DDOS, it would bring some internet traffic when attacker access to the internet. We find the attack signatures can describe the attack, but we cannot find the order of launching attack because of the orders in UDP packet. However, it cannot hinder us to extract signatures to describe TCP traffic.

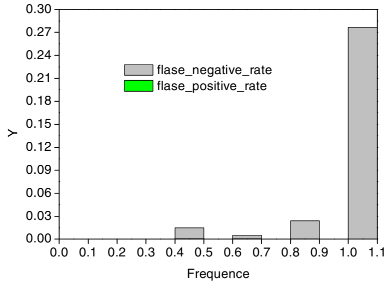


Fig. 7. The result of feature extraction experiment by using cluster0 from Phase-5-dump-inside dataset

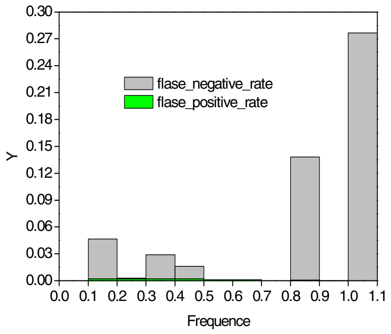


Fig. 8. The result of feature extraction experiment by using cluster1 from Phase-5-dump-inside dataset

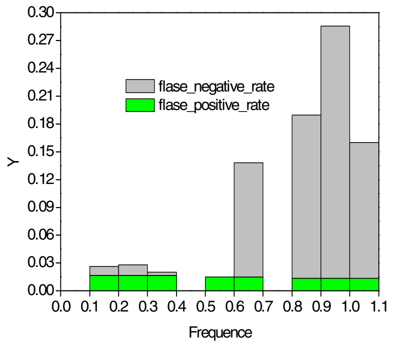


Fig. 9. The result of feature extraction experiment by using cluster2 from Phase-5-dump-inside dataset

Table 7. the attack signature of phase-4-dump-inside dataset

sample	the frequency range	attack signature
cluster0	[0.6,0.7)	✓ l/User-Agent: Mozilla/3.01 (Win95; I;)Host: www.af.milAccept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */* …
cluster1	[0.5,0.6) [0.6,0.7)	✓ HTTP/1.0Referer: http://www.af.mil/User-Agent: Mozilla/2.0 (compatible; MSIE/3.01; Windows 95)Host: www.af.milAccept: image/gif, image/x-xbitmap,image/jpeg, image/pjpeg, */*Accept-Language: enUA-pixels: 1024x768UA-color: color32UA-OS: Windows 95UA-CPU: i686 …
The fourth stage cluster2	[0.3,0.4)	✓ jpg HTTP/1.0Referer: http://www.af.mil/User-Agent: Mozilla/2.0 (compatible; MSIE/3.01; Windows 95)Host: www.af.milAccept: image/gif, image/x-xbitmap,image/jpeg, image/pjpeg, */*Accept-Language: enUA-pixels: 1024x768UA-color: color32 UA-OS: Windows 95UA-CPU: i686 …

5 Conclusion

This paper presents a combining supervised and unsupervised learning for automatic attack signature generation system model, which based on feature from the transport and network layer. These features are more resilient to payload encryption. Our model deals with the packet from internet and generates attack signature. Experiment results show that our work can extract the effective signature. For future work, we plan to improve the effectiveness of abnormal flow clustering module. We will extend the formulation to an online learning setting.

Acknowledgments. This work is supported by National Natural Science Foundation of China under Grant No.61202495.

References

1. China Internet Network Information Center. China Internet Development Statistics Report, http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201403/P020140305346585959798.pdf
2. Wang, X.L.: Analysis and Detection of Botnet Anomaly Traffic. Beijing University of Posts and Telecommunications. Ph D Thesis, Beijing (2011)
3. Niu, S.Z.: Introduction to Secure Information Systems, pp. 3-15. Beijing University of posts and telecommunications Press, Beijing (2004)
4. Tang, Y., Lu, X.C., Wang, Y.J.: Survey of Automatic Attack Signature Generation. Journal on Communications 30, 96–105 (2009)
5. Kreibich, C., Crowcroft, J.: Honeycomb-creating intrusion detection signatures using honeypots. In: Proceedings of the Second Workshop on Hot Topics in Networks, Boston, pp. 51–56 (2003)
6. Tang, Y.: Research on Network-based Automatic Attack Signature Generation. National University of Defence Technology. Ph D Thesis, Changsha (2008)
7. Wang, K., Cretu, G.F., Stolfo, S.J.: Anomalous payload-based worm detection and signature generation. In: Valdes, A., Zamboni, D. (eds.) RAID 2005. LNCS, vol. 3858, pp. 227–246. Springer, Heidelberg (2006)
8. Vargiya, R., Chan, P.K.: Boundary detection in tokenizing network application payload for anomaly detection. In: Proceedings of ICDM Workshop on Data Mining for Computer Security (2003)
9. Comar, P.M., Liu, L.: Combining Supervised and Unsupervised Learning for Zero-Day Malware Detection. In: Proceedings IEEE INFOCOM, pp. 2022–2030. IEEE Press (2013)
10. Han, J.W., Kamber, M.: Data Mining Concepts and Techniques, pp. 211–321. China Machine Press, Beijing (2011)
11. Lincoln Laboratory, DARPA Intrusion Detection Scenario Specific Data Sets (2000), http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000data.html

The Study on the Increasing Strategy of Detecting Moving Target in Wireless Sensor Networks

Jialong Xu[1], Zhigang Chen[1], Anfeng Liu[2], and Hong Song[2]

[1] School of Software, Central South University, China
[2] School of Information Science and Engineering, Central South University, China
qxjl1010@gmail.com, {czg,afengliu,songhong}@csu.edu.cn

Abstract. Wireless Sensor Networks (WSNs) have promising and valuable applications. More and more researchers have conducted research in this area. Because sensor nodes are small and cheap, they usually use battery to provide energy. However, as a great number of nodes are arranged in the network and the working environment is bad, it is difficult to replace batteries or recharge. How to make full use of network resources to ensure the high quality of the network and achieve a balanced network is a problem to be faced. This paper focuses on improvement in wireless sensor networks by improving the MAC protocol.

MAC protocols are basic protocols in WSNs and they influence the performance of network a lot. In those networks that need to track mobile targets, how to achieve the goal of increasing the monitor rate and balance the consumption of network are two very important objectives focused by modifying the MAC protocols. In this paper, we proposed the IMMA to improve the efficiency of the network. The core of this algorithm is by changing the cycle of the state to balance the energy consumption, and increase the monitor rate. IMMA balances energy consumption and enhances the monitor rate of the moving targets. Theoretical and experimental results show that the algorithm improved network efficiency a lot.

Keywords: MAC algorithm, network energy balance, objective monitoring rate.

1 Introduction

The technology of wireless sensor network (WSN) includes a great number of sub-technologies, such as wireless communication technology, embedded technology, micro-electromechanical systems, system on chip etc. WSN has much more advantages than older network, it has lower cost and lower power consumption, does not need to be distributed, self-organizing etc. WSN is composed of many sensor nodes and one or several base station, it usually communicates by radio communication. Nowadays, WSN plays an important role in many fields such as in Bio Medical, environmental monitoring, national security etc.

WSNs are always required to run a long time. However, the node in WSNs has limited energy and it cannot be charged or replaces the battery, so the problem of energy consumption of sensor nodes has become a focus problem. There are numerous studies tried to balance the energy consumption in all nodes in the network and prolong the network lifetime, eventually improve the efficiency of the network. But when part of the network dead, work efficiency of the network will inevitable drop. Therefore, balancing the whole energy consumption in WSNs has been seemed as the best solution.

So far, there are many researchers studied on the energy consumption in WSNs [1-3]. The emphases on those studies are different, despite they are all focus on energy consumption. In all of the studies, routing algorithm and network topology can do the most to improve. There are a great number of research productions in those two aspects such as the design of routing protocol based on energy consumption [4, 5], the design of network topology based on energy consumption [3] and the design of MAC protocol based on energy consumption etc [6-8]. Some researchers divided network to several rings [9], they analyzed the data bits in nodes of each ring, and then come up with a formula between data carrying capacity and the node density in each ring. Experiments showed that, this algorithm can balance the energy consumption in WSNs.

In order to increase the monitoring rate and balance the energy consumption in the network, in this paper, we present a strategy without changing the original condition in WSNs, such as the sensor node numbers, the hardware equipment of the sensor nodes or the hardware equipment of the base station. In this strategy, the nodes have three different conditions and the time periods in different conditions are changed according to the distance from base station. We presented the IMMA to distribute more monitoring task to those nodes which have less transporting task.

The remainder of this paper is structured as follows. In section 2, we introduce the system model. In section 3, we present the IMMA and demonstrate its efficiency. In section 4, we present the simulation results. We give our conclusions and future work in section 5.

2 System Model

2.1 The Network Model

As illustrated in Fig.1, the study was studied in a round network [10]. We assumed that all nodes are deployed in the two-dimensional circular region and sensor nodes are deployed randomly. The sink was deployed in the center of the circle. There is a moving target randomly deployed in the network.

The sensor nodes have the following properties:

1. All of the nodes have the same initial energy reserve and they cannot move after deployed.
2. Nodes have the same transmission distance, all nodes can transmit data.
3. Nodes are divided into three states: active state, monitoring state and sleep state. Active nodes can transmit data and sense the target, monitoring nodes can only transmit data, and sleep nodes can do nothing. Each nodes switch between three states automatically.

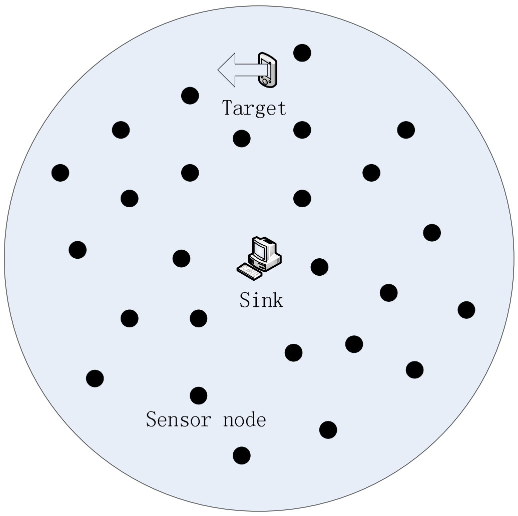


Fig. 1. The network model

The moving target can be perceived by active nodes, and it would moved randomly by a certainly speed. The moving target cannot move out of the network.

In our network model, the energy consumption of nodes usually through the following operations: monitoring whether moving target is presence nearby; transmit a data packet; receive a data packet; listen to the radio signal, etc. In this paper, we mainly consider the energy consumption in data transmit and monitoring. We divide energy consumption into two parts: the first parts are the energy consumption in transport data and receive data, we call that consumption the transmit energy consumption. The second part is the energy consumption in monitoring the moving target:

$$E_{total} = E_{transport} + E_{receive} + E_{sense} \tag{1}$$

2.2 The Transmission Energy Model

We must present the relation between nodes transmission radius and the actual transmission distance. If one node's actual transmission distance is d, then the transmission distance to the sink of this node is:

$$r = df(\rho)\cos\sigma \tag{2}$$

f(ρ) is the function of the node density, σ is the angle between the actual transmission path and the radial line. We may understand the number of data packets each node may take [11]:

$$S_x = (z+1) + \frac{z(1+z)r}{2l} \tag{3}$$

We use the typical energy consumption model [12]:

$$\begin{cases} E_{transport} = lE_{elec} + l\varepsilon_{fs}d^2 & if\ d < d_0 \\ E_{transport} = lE_{elec} + l\varepsilon_{amp}d^4 & if\ d \geq d_0 \end{cases} \tag{4}$$

$$E_{receive} = lE_{elec} \tag{5}$$

E_{elec} is the energy consumption when nodes sending data. When the node transmission distance is less than the threshold value d_0, we use the free space model. When the node transmission distance is more than d_0, we use multi path fading model. ε_{fs} and ε_{amp} are the energy consumptions when nodes amplify the power. L is the bit number [12].

Table 1. The parameter of nodes transmission

Parameter	Value
Threshold distance(d_0)(m)	87
Sensing range γ_s(m)	15
$E_{elec}(nJ/bit)$	50
$\varepsilon_{fs}(pJ/bit/m^2)$	10
$\varepsilon_{amp}(pJ/bit/m^2)$	0.0013
Initial energy(J)	0.5
$\alpha(nJ/bit)$	50
l(Mbps)	1

2.3 The Sensing Energy Consumption Model

Nodes have three states: the first state is sleep state. At this time, nodes do nothing; neither transmits data nor sense the target:

$$E_{sleep} = P_{slp} * t \tag{6}$$

The second state is monitoring state. At this time, nodes do some transmit missions, but do not sense the target:

$$E_{listen} = P_{lst} * t \tag{7}$$

The third state is active state. At this time, nodes can transmit data and sense the target:

$$E_{active} = P_{rcv} * t_{rcv} + P_{lst} * t_{lst} \tag{8}$$

P_{rcv} is the power when nodes in the active state, P_{lst} is the power when nodes in the monitoring state [13].

Table.2 The parameter of monitoring target

Parameter	Value
$P_{rcv}(mW)$	45
$P_{lst}(mW)$	45
$P_{slp}(\mu mW)$	90

3 The Research of IMMA Strategy

In the target monitoring network, no matter what strategies we choose, we may lost the target sometimes, because the target always moved and not all of nodes stay in active state. Besides, with the demand for the target monitoring accuracy improved, we need more active nodes to monitor the target. If we cannot improve the protocol, the monitoring accuracy will decrease apparently.

On the other hand, there usually some nodes need to do more transmission task than other nodes, which may cause those nodes used up their energy faster than others. When some nodes used up their energy, it may cause energy hole. The energy holes become bigger and bigger, the network will die. However, when network died, some nodes may remain lots of energy because they take less transmission task than other nodes. Then, that energy will be wasted.

In this section, we improve the media access control algorithm; the main feature is to gradually increase the monitoring task in those nodes which have less transmission task. This strategy may increase the monitoring accuracy and avoid the energy wasting.

As illustrated in Fig.2, we consider the network as many rings nested each other. The radius of the network is R; the ring width is the transmission radius is d; the distance between the ring and sink is r_c:

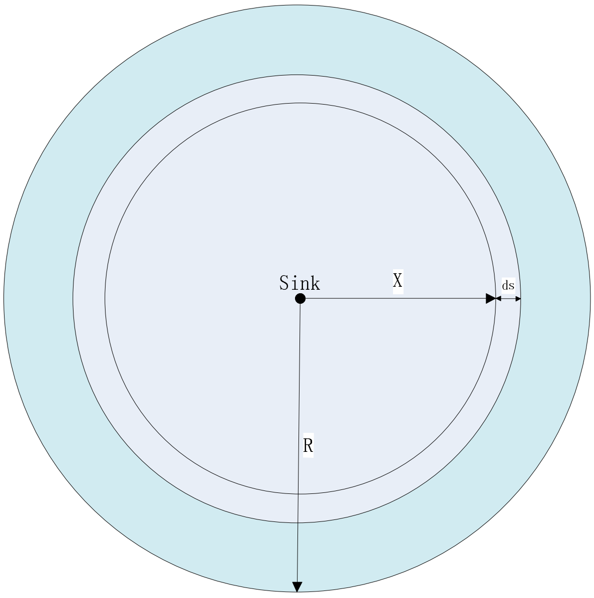


Fig. 2. The strategy of node transmission

3.1 The Analysis of Energy Consumption of Sensor Nodes to Transmit and Receive Data

We assume there are M nodes in the network, the nodes density is:

$$\rho = \frac{M}{\pi R^2} \tag{9}$$

The nodes numbers in the ring which distance from sink is r_c is:

$$\rho \pi d(2r_c + d) \tag{10}$$

Sensor nodes in each ring not only transmit and receive data which comes inside the ring, but also need to transmit the data comes out of the ring. So the data in the ring is:

$$\pi\rho(R^2 - r_c^2) \tag{11}$$

From (10) and (11), we may understand the amount of data each node needs to undertake is:

$$\frac{\pi\rho l(R^2 - r_c^2)}{\pi\rho d(2r_c + d)} = \frac{l(R^2 - r_c^2)}{d(2r_c + d)} \tag{12}$$

The transmission energy consumption in the node distance from sink is r_c is:

1) When the node transmission distance is less than the threshold value d_0:

$$\begin{cases} \frac{l(R^2 - r_c^2)}{d(2r_c + d)}\left(E_{elec} + \varepsilon_{fs} d^2\right), r_c \geq d \\ \frac{l(R^2 - r_c^2)}{d(2r_c + d)}\left(E_{elec} + \varepsilon_{fs} r_c^2\right), r_c < d \end{cases} \tag{13}$$

2) When the node transmission distance is more than the threshold value d_0:

$$\begin{cases} \frac{l(R^2-r_c^2)}{d(2r_c+d)}\left(E_{elec}+\varepsilon_{amp}d^4\right), r_c \geq d \\ \frac{l(R^2-r_c^2)}{d(2r_c+d)}\left(E_{elec}+\varepsilon_{amp}r_c^4\right), d_0 \leq r_c < d \\ \frac{l(R^2-r_c^2)}{d(2r_c+d)}\left(E_{elec}+\varepsilon_{fs}r_c^2\right), r_c < d_0 \end{cases} \quad (14)$$

As illustrated in Fig.3, in the network radius is 200 meters, we can see the nodes data bits is:

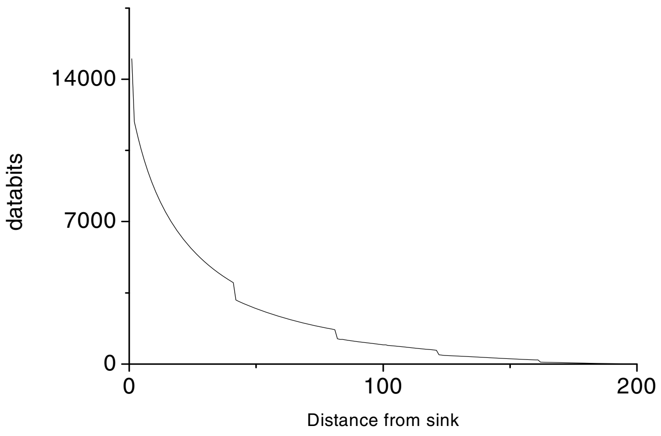


Fig. 3. The amount of data in theory

As illustrated in Fig.4, in the network radius is 200 meters, we can see the nodes transmission energy consume is:

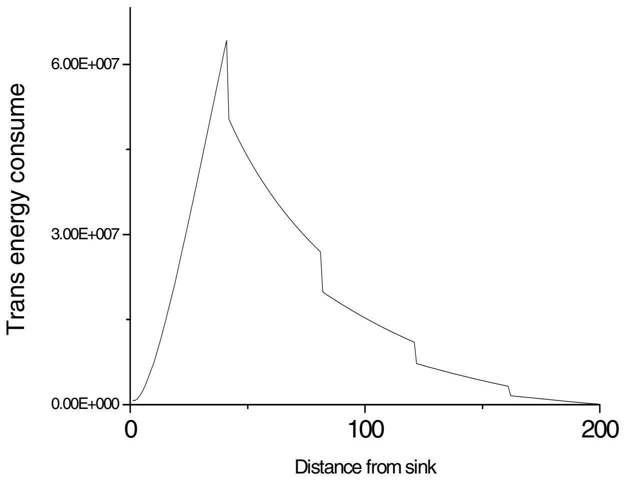


Fig. 4. The energy consumption in theory

From Fig.4, we can see the transmission energy consumption can be roughly divided into two parts. When the node distance from sink is less than its transmission distance, the energy consumption is increased. On the contrary, when the node distance from sink is more than its transmission distance, the energy consumption is decreased.

Definition 1. The area which distance from sink is equal to the nodes' transmits distance we may call it limit ring.

We call those areas which distance from sink is less than limit ring inner ring; we call those nodes in the inner rings inner node. We call the areas which distance from sink is more than limit ring outer ring; we call the nodes in the outer rings outer node. In the inner rings, the active time gradually increased inward, we use k_{inner} to present the increase coefficient. In the outer rings, the active time gradually increased outward, we use $k_{external}$ to present the increase coefficient.

3.2 The Design of IMMA

According to previously known, in the WSNs, energy consumption is the biggest problem restricting the network efficiency. The sensor nodes on the limit ring are very easy to deplete their energy, which may cause the network death. If the node active state, monitoring state and sleep state is transformed by the ratio of 1:1:1, it may inevitable lead to the network energy waste. So how to balance the transmission consumption and the monitoring energy consumption is the key issue of our research.

We present IMMA, it based on the distance from sink adjust the time which nodes in the active state and monitoring state.

```
1.  the nodes start work
2.  set the initial state to all nodes randomly
3.  calculate the distance from sink
4.  **IF** the node is in the outer ring
5.  the nodes active time and monitoring time increase by $k_{external}$
6.  **ELSE IF** the node is in the inner ring
7.  the nodes active time and monitoring time increase by $k_{inner}$
8.  **ELSE IF** the node is in the limit ring
9.  The three states set to 1:1:1
10. The node changes the state automatically in the network   period
```

3.3 The Analysis of Node Status Cycle

IMMA focuses on achieving energy balance and improving the monitoring accuracy by adjusting the nodes time in different states. Sensor nodes continually change state between active states, monitoring states and sleep states:

$$T_{cycle} = T_{active} + T_{monitoring} + T_{sleep} \tag{15}$$

T_{cycle} is the nodes totally cycle time, T_{active} is the time that nodes in the active state, $T_{monitoring}$ is the time that nodes in the monitoring state, T_{sleep} is the time that nodes in the sleep state. T_{cycle} in all of the nodes are the same in the network. However, the time between each state are vary.
The three states of the nodes at the limit ring are equal:

$$T_{active}:T_{monitoring}:T_{sleep} = 1:1:1 \tag{16}$$

The ratio of three states of the nodes in the inner ring is:

$$T_{active}:T_{monitoring}:T_{sleep} = 1:\frac{1}{k_{inner}^{d-r_C+1}}:\frac{1}{k_{inner}^{d-r_C+1}} \tag{17}$$

The ratio of three states of the nodes in the outer ring is:

$$T_{active}:T_{monitoring}:T_{sleep} = 1:\frac{1}{k_{external}^{r_C-d+1}}:\frac{1}{k_{external}^{r_C-d+1}} \tag{18}$$

As illustrate in Fig.5, the nodes effect in the network is:

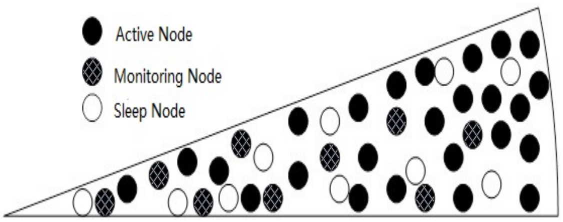


Fig. 5. The network effect

After we used IMMA, the network efficiency improved apparently. We will present the simulation results in section 4.

4 Simulation

4.1 Simulation Setup

We use OMNet4.0++ to do simulation experiment. The network operating in a circle that radius is 200 meters. Each node has 0.5J initial energy, the nodes transmission radius is 40 meters and the monitoring radius is 20 meters. There are 500 nodes in the network.

In order to test the efficiency of IMMA, we compare IMMA with normal algorithm which three states are equal.

4.2 The Analysis of IMMA

The statistical analysis of the transmission energy consumption illustrated in Fig.6. The experimental results and theoretical analysis showed the same trend, the transmission energy consumption of nodes in the limit ring will reach the maximum:

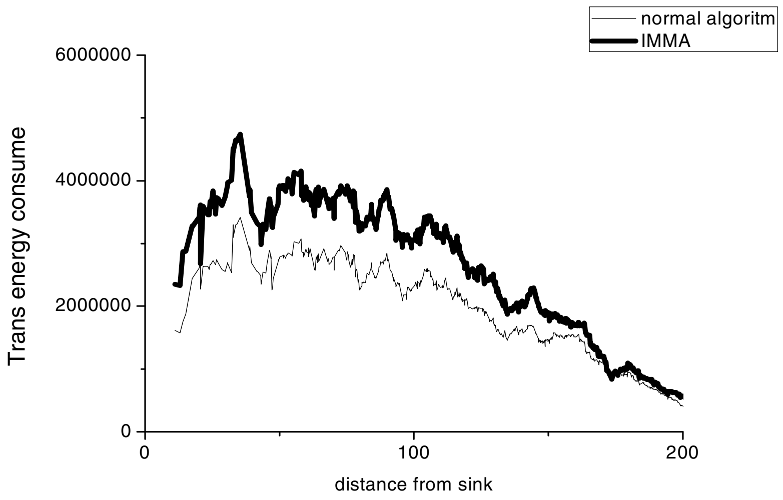


Fig. 6. 1 Transmission energy consumption

Next we tested the effectiveness of IMMA. As illustrate in Fig.7, after we used the IMMA, the nodes states changed apparently:

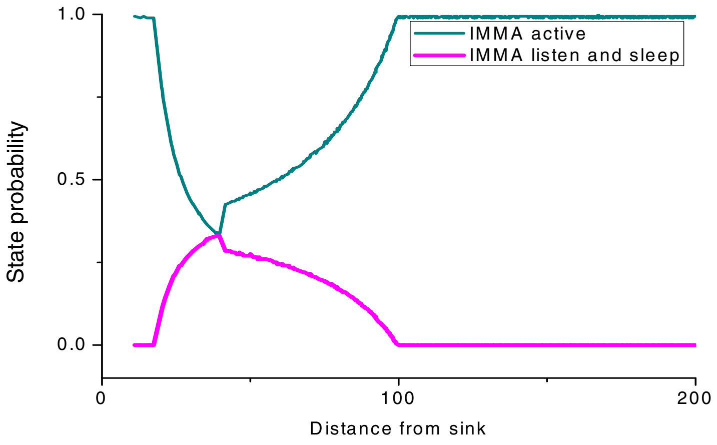


Fig. 7. The nodes states

Then, the monitoring energy consumption increased as illustrate in Fig.8:

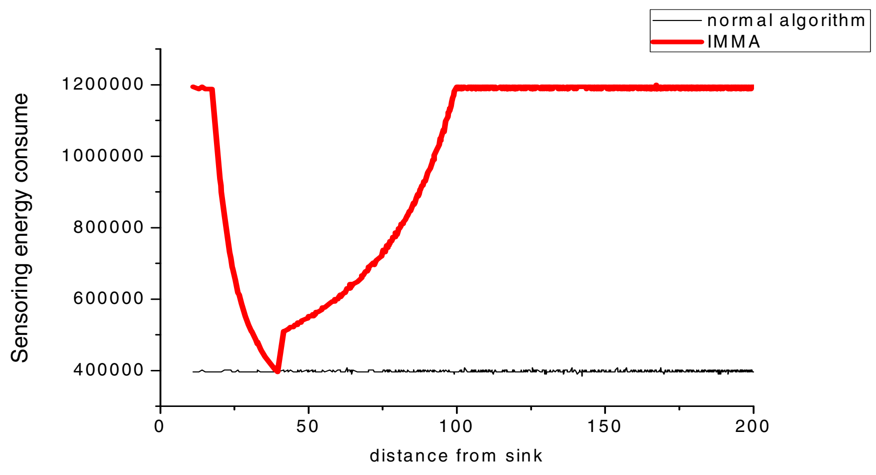


Fig. 8. Monitoring energy consumption

4.3 The Remaining Energy Analysis

As illustrate in Fig.9, the network using IMMA dead after 637 rounds, the network do not using IMMA dead after 641 rounds. The normal algorithm will waste lots of energy because when network dead, some sensor nodes in the network remained a lot of energy. On the contrary, IMMA can used up that energy to monitor target.

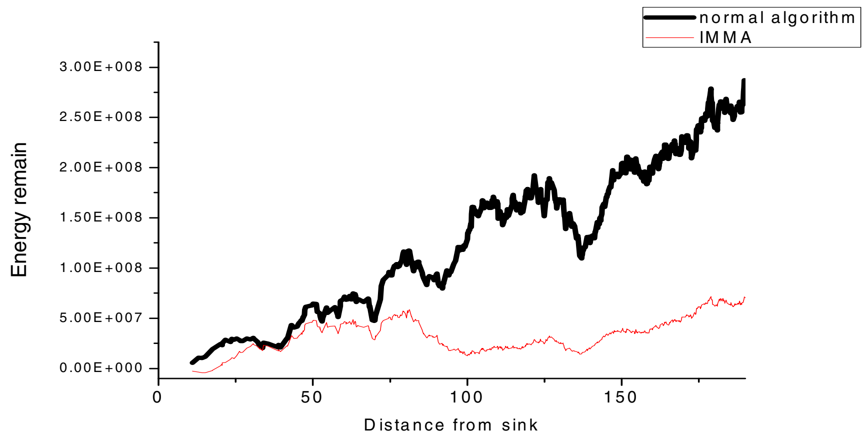


Fig. 9. The remaining energy

4.4 The Target Monitoring Rate Analysis

In order to test the efficiency of IMMA, we analyzed the target monitoring accuracy. We used Centroid localization algorithm to locate the moving target. We design two experiments; the first network deployed 500 nodes, the second network deployed 800 nodes. As illustrated in Fig.10, the monitoring accuracy of the first network increased 60%, the monitoring accuracy of the second network increased 40%. We can see IMMA can do better in the nodes arranged loose network.

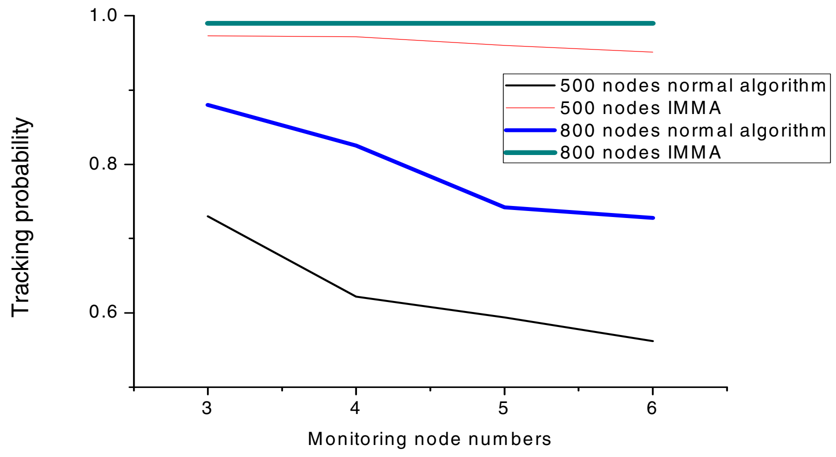


Fig. 10. The monitoring accuracy

5 Conclusion

This paper studied by change the nodes state time to balance the energy consumption and increase target monitoring accuracy. The energy in WSNs is limited, how to efficiently use the limited energy is particularly important. In order to solve this problem, we presented IMMA, which gradually increase the active state and monitoring state in the inner ring and the outer ring. Theoretical analysis and simulation results confirmed IMMA can improve the target monitoring accuracy and balance the energy consumption.

Acknowledgement. This paper is supported in part by National Natural Science Foundation of China under Grant No.61202495.

References

1. Li, J.Z., Li, J.B., Shi, S.F.: Concepts, issues and advance of sensor networks and data management of sensor networks. Journal of software 14(10), 1717–1727 (2003)

2. Shih, E., Cho, S.H., Ickes, N., et al.: Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, pp. 272–287. ACM (2001)
3. Salhieh, A., Weinmann, J., Kochhal, M., et al.: Power efficient topologies for wireless sensor networks. In: 2001 International Conference on Parallel Processing, pp. 156–163. IEEE (2001)
4. Tang, Y., Zhou, M., Zhang, X.: Overview of routing protocols in wireless sensor networks. Journal of Software (2006)
5. Shah, R.C., Rabaey, J.M.: Energy aware routing for low energy ad hoc sensor networks. In: Wireless Communications and Networking Conference, vol. 1, pp. 350–355. IEEE (2002)
6. Haapola, J., Shelby, Z., Pomalaza-Ráez, C., et al.: Multihop medium access control for WSNs: an energy analysis model. EURASIP Journal on Wireless Communications and Networking, 523–540 (2005)
7. Ye, W., Heinemann, J., Estrin, D.: An energy-efficient MAC protocol for wireless sensor networks. In: INFOCOM 2002. Proceedings of the Twenty-First Annual Joint Conferences of the IEEE Computer and Communications Societies, vol. 3, pp. 1567–1576. IEEE (2002)
8. Van Dam, T., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp. 171–180. ACM (2003)
9. Olariu, S., Stojmenovic, I.: Data-centric protocols for wireless sensor networks. In: Stojmenovic, I. (ed.) Handbook of Sensor Networks: Algorithms and Architectures, pp. 417–456. WiIey (2005)
10. Yanjie, H.: Research on routing protocols for wireless sensor networkenergy consumption balance. Wuhan University of Technology (2010)
11. Chiang, M., Low, S.H., Calderbank, A.R.: Layering as optimization decomposition: A mathematical theory of network architectures. Proc of the IEEE 95(1), 255–312 (2007)
12. Zhang, C.-I., Chen, Z.-G., Liu, A.-F.: Energy Hole Avoidance Strategy for Nonuniform Distribution Wireeless Sensor Network. Computer Engineering 36(2), 83–86 (2010)
13. Haapola, J., Shelby, Z., Pomalaza-Ráez, C., et al.: Multihop medium access control for WSNs: an energy analysis model. EURASIP Journal on Wireless Communications and Networking (4), 523–540 (2005)

A *CRC*-Based Lightweight Authentication Protocol for EPCglobal Class-1 Gen-2 Tags

Zhicai Shi[1,2], Yongxiang Xia[1], Yu Zhang[3], Yihan Wang[1], and Jian Dai[1]

[1] School of Electronic&Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, P.R. China
`{szc1964,x-free}@163.com, e-han@sues.edu.cn, 1101196001@qq.com`
[2] Department of Computing, Macquarie University, Sydney 2109, Australia
[3] Sino-Korean School of Multi-media, Shanghai University of Engineering Science, Shanghai 201620, P.R. China
`zhangyu43321@163.com`

Abstract. Authentication is one of the most important technologies to protect the privacy and security of RFID systems. The EPCglobal Class-1 Gen-2 specification is an important standard for RFID. The tags conforming to this standard have limited computing and store resources, and no more attentions are paid to their security and privacy. So the application of these tags is not secure. But this kind of tags is considered as the main stream for RFID applications. In order to solve the secure problems for this kind of RFID tags we propose a lightweight authentication protocol based on *CRC*, which can assure forward security and prevent information leakage, location tracing, eavesdropping, DOS attack, replay attack, and spoofing. This protocol avoids *CRC* collision by reasonably dividing the tag's identification information into two different parts and it enhances the mutual authentication strength between tags and readers by dual authentications. This protocol only uses the computing resources embedded in tags and it is very suitable to low-cost RFID systems.

Keywords: RFID, authentication protocol, CRC, privacy, security.

1 Introduction

With the development of Internet of Things, Radio Frequency IDentification(RFID) technique gets the broad attention. RFID is a pervasive technology deployed in everyday life in order to identify objects using radio-waves, without visible light and physical contact. Today, RFID systems have been successfully applied to manufacturing, supply chain management, agriculture, transportation, healthcare, electronic-payment, e-passport and other fields[1]. But, the wide applications of RFID into modern society may make the security and privacy of consumers exposed to threats and risks. For example, businesses may have malicious competitors to collect unprotected RFID information, use forgery tags to provide some wrong information, or even launch denial of service attacks against RFID systems. On the other hand, as a consumer, it is naturally preferred that the information of his RFID-tagged products should be private and secure. However, a tag reader can read the content of an un-protected tag, tracing the RFID-tagged product and even identifying the person carrying the tagged product. To protect the private information on the RFID tags, some special techniques can be used to prevent malicious readers from accessing the tags. Currently, these techniques are divided into two main categories: physical approaches, encryption mechanism and protocol. Further research results indicate encryption mechanism and protocol is a more flexible and effective approach for ensuring the security and privacy of RFID systems. RFID authentication is a special encryption protocol which is widely deployed. Now, many authentication protocols for RFID systems have been proposed. Some protocols use the complicated encryption algorithms and they are not suitable for the EPCglobal Class-1 GEN-2 RFID specification (which is called the C-1 G-2 RFID specification for short in this paper). This specification is one of the most important standards proposed by EPCglobal and it can be considered as a “universal” standard for low-cost RFID tags. The C-1 G-2 RFID tags are very cheap and their effective transmitting range is about 2 to 10 meters. It is believed that the C-1 G-2 RFID tags will become the mainstream for developing RFID systems[2]. But the C-1 G-2 RFID tags only have the limited computing and store resources and they pay little attention to the security and privacy threats. So it is very necessary to design a lightweight authentication protocol suitable for the C-1 G-2 RFID tags.

2 The RFID System and The EPCglobal Class-1 GEN-2 RFID Specification

A RFID system consists of three components: Radio Frequency(RF) tags, RF readers and a backend server[3], as shown in Fig.1. A tag is basically a silicon chip with antenna and a small memory that stores its unique identifier known as EPC (Electronic Product Code). A reader is a device capable of sending and receiving data in the form of radio frequency signal. This device is used to read EPC from the tag. A backend server is used to store the information related to the objects being tagged with the RFID tags and cooperates with readers to finish some complicated functions. The basic working procedure for an RFID System is that the objects are tagged and the tags store the related data of the tagged objects. The tag receives the query from the reader and transmits data stored in it to the reader. The reader transfers the data to the backend server through wired or wireless networks. The reader could be fixed as well as mobile. The server processes the request from the reader and sends the related information about the tagged object to the reader.

For an RFID system, a tag is a special device. Its computing and memory resource is very limited. There are two main types of tags: active tag and passive tag. Active tags include miniature batteries used to power the tag and they are capable to transmit data over longer distance. Another is passive tag which has no any battery in it, it needs to be activated by the RF signal beamed from the reader. Passive tags are smaller, less expensive and used for a shorter range. Because of their priority this kind of tags are applied widely. Our researches mainly focus on secure and private problems on this kind of low-cost Tags. During researching we also note that since well-designed conventional cryptographic protocols can be effectively implemented on resource-abundant backend servers and readers, it is usually assumed that the channels between backend servers and readers are secure. However, because of the limited resource in tags it has to assume that the channel between tags and readers is insecure. Readers have electric power enough to transmit signals over longer distance and tags only have limited electric energy to transmit signals over shorter distance. So the communication channels between readers and tags are asymmetric. We call the channel from readers to tags as forward channel and the channel from tags to readers as backward channel. These two channels are open and insecure. Most attacks to RFID systems are resulted from these insecure channels. These attacks include information leakage, eavesdropping, location tracing, forward security, desynchronization, replay attack, and spoofing[4].

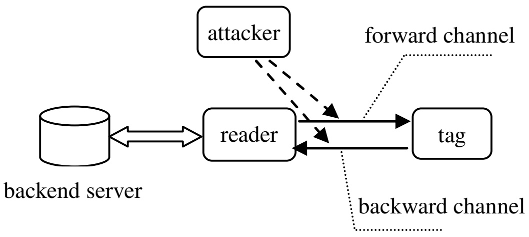


Fig. 1. The component of an RFID system

A C-1 G-2 RFID tag is a passive tag, and its power is triggered by the readers. Cost limitation and limited resources dictate that C-1 G-2 tags cannot afford the cost expensive public key encryptions, symmetric encryption, or even hash functions. A C-1 G-2 RFID tag supports a 16-bit Pseudo-Random Number Generator (PRNG), and a 16-bit Cyclic Redundancy Code (CRC) checksum. The latter is used to detect error in transmitted data. Tag memory is insecure and susceptible to physical attacks. Tags cannot be trusted to store global, long-term secrets. A kill command with a 32-bit PIN is used to protect the privacy of a C-1 G-2 RFID tag by permanently making it disabled. A 32-bit access PIN is required to trigger a tag into the secure mode. Then the tag is allowed to READ/WRITE. Readers are assumed to have a secure connection to a backend database. Some research works[2,5] pointed out several weaknesses of the C-1 G-2 RFID specification as follows:

(1) The C-1 G-2 RFID tags use the kill command to permanently make a tag unusable to protect the privacy of the tags. This approach is not appropriate for many occasions. For example, when buying a product the customer needs the information of the product for warranty purpose, but the tag on the product have being killed after its purchase.

(2) It is feasible for an attacker to derive the PIN by eavesdropping the communication between tags and readers. After deriving the PIN, it is easy for the attacker to access and trace the tags. So the C-1 G-2 RFID tags can be deployed widely only if their security and privacy problems are solved.

3 The Related Works

Authentication is the process of ensuring that the users are the persons whom they claim to be. Therefore, the goal of authentication is that the tag authenticates the reader before it is accessed and the authorized readers can get the content of the valid tags. Moreover, the private information about the tag would not be leaked to un-authorized entities.

An RFID authentication protocol is a special cryptographic protocol, where resource-limited RFID tags are involved. This kind of protocol is called as the lightweight authentication protocol. For this case, conventional authentication protocols that concern symmetric key computations or even public key computations are not applicable.

Many research works have been done for RFID authentication and some authentication protocols use encryption schemes or hash function to solve most of the security and privacy problems of RFID systems. But they either cannot conform to the C-1 G-2 RFID specification or suffer from some security flaws. Only a few proposed schemes can be implemented on C-1 G-2 RFID tags. Unfortunately, these schemes still suffer from some security weaknesses[5].

S. Karthikeyan and M. Nesterenko[6] proposed an authentication protocol for the C-1 G-2 RFID specification based on simple *XOR* operation and matrix calculations. Initially, two matrices $M1$ and $M2^{-1}$ are stored on each tag, and two matrices $M2$ and $M1^{-1}$ are stored on the reader or the backend server, where all matrices are size $p \times p$, and $M1^{-1}$ and $M2^{-1}$ are the inverses of $M1$ and $M2$ respectively. The tag and the reader also store a key K which is a vector of size q, where $q=r \times p$. When the reader inquiries a tag, the tag computes $X=KM1$, and sends X to the reader. Then the reader forwards X to the backend server, where the server will search its database. If it can find a match, then the tag is identified, and the server renews the key. The server first computes $Y= (K1 \oplus K2\ \oplus \ldots \oplus Kr)M2$, randomly selects a vector X_{new} of size q, computes $K_{new}=X_{new}M1^{-1}$ and $Z=K_{new}M2$, and sends (Y, Z) to the reader, which forwards (Y, Z) to the tag. Upon receiving the response from the reader, the tag verifies whether the equation $YM2^{-1}$ is equal to $K1$□$K2$□......□Kr. if they are equal, the tag permits to be accessed by the reader and updates the key as $K_{new}=ZM2^{-1}$, where the tag does not check whether Z is legal or confident. Therefore, an attacker can replace the transmitted Z with a faked value Z^*. Upon receiving a valid Y and the faked Z^*, the tag will authenticate Y successfully and then will update the key. The reader and the tag cannot authenticate each other any more since the key is wrongly updated. Hence the desynchronization or DOS attack occurs. Once the key of the tag is updated by a faked value Z^*, the attacker can replay the eavesdropped messages many times so that the replay attack occurs. Besides, the attacker can trace the tag by repeating to send the eavesdropped messages to the tag[5].

For the C-1 G-2 RFID specification, D. N. Duc et al.[7] used the pseudorandom function $f(\ )$ and the *CRC* function $CRC(\ )$ provided by the tag to propose an authentication protocol. Initially, each tag and the backend server share the tag's EPC code, the tag's access PIN, and an initial key $K0$. In order to enhance the security of the exchanged messages the key will be updated after each successful authentication, and

Ki denotes the key after i^{th} authentication. But the tag or the backend server updates its key only if they receive the message "end session" from the reader. If one among them does not receive the ending message it cannot update its key value, so that their key is different. Hence they cannot authenticate each other any more and desynchronization attack or DOS attack occurs. If the session ending message to the server is intercepted, then the server will hold the old key; therefore, a counterfeiting tag can replay the old message (*M1, r, C*) to disguise as a legitimate tag to spoof the reader. So, the scheme cannot prevent spoofing attack. Besides, this authentication protocol cannot provide forward secrecy. Suppose a tag is compromised, then the attacker would get the values (EPC, PIN, *Ki*) of the tag. By eavesdropping the previous messages(e.g. *M1, M2, r*), the attacker can judge whether these messages come from the same tag by performing the following operations. For each eavesdropped message (*M1, M2, r*), the attacker computes $M1 \oplus M2$ to derive the value $CRC(\mathrm{EPC} \oplus r) \oplus CRC(\mathrm{EPC} \| \mathrm{PIN} \| r)$, and then, using the compromised values (EPC, PIN, *Ki*) and the eavesdropped *r*, the attacker can do the same computation to verify whether they came from the same tag so that a compromised tag can be traced[5].

Chien et al.[5] analyzed some previous similar authentication protocols and then proposed a mutual authentication protocol for RFID tags conforming to the C-1 G-2 RFID standard. This protocol has gotten wide attention and analysis. Their proposed scheme consists of two phases: the initialization phase and the authentication phase. In the initialization phase, each tag is denoted as *Tagx* and it is identified by *EPCx*. Each tag stores its *EPCx*, its initial authentication key $K_{x\text{-}0}$ and its initial access key $P_{x\text{-}0}$. The authentication key and the access key will be updated after each successful authentication, and the authentication key after the i^{th} successful authentication is denoted as $K_{x\text{-}i}$ and the access key after the i^{th} authentication is denoted as $P_{x\text{-}i}$. For each tag, the server also maintains a record of six values in its database.(1) *EPCx*. (2) K_{old} denotes the old authentication key, and it is initially set to $K_{x\text{-}0}$. (3) P_{old} denotes the old access key, and it is initially set to $P_{x\text{-}0}$; (4) K_{new} denotes the new authentication key, and it is initially set to $K_{x\text{-}0}$, too. (5) P_{new} denotes the new access key, and it is initially set to $P_{x\text{-}0}$, too. (6) DATA denotes all the other information about the tagged object. The design of two sets of authentication key and access key is to defend DOS attack that causes out of synchronization between the tag and the server. The scheme claimed it can assure forward secrecy, resist the replay attack and being traced. But Pedro Peris-Lopez et al.[8] found the protocol proposed by Chien et al. has some secure flaws and for this protocol there maybe exist counterfeiting attack of a tag or a reader and desynchronization attack. Daewan Han and Daesung Kwon[9] pointed out that *CRC* is a linear function and the inherent vulnerability of *CRC* functions results in the secure flaws of the protocol proposed by Chien et al.. They also pointed out that an attacker can impersonate a valid tag temporarily by a single eavesdropping and he can clone a valid tag by eavesdropping two consecutive sessions. Finally, they deduced that Chien et al.'s protocol cannot enhance the RFID security any more than the original EPC standard[10]. But they have not proposed any better authentication method.

Hu Tao and Wei Guoheng[11] proposed an anonymous bidirectional RFID authentication protocol for the C-1 G-2 RFID tags. This scheme was designed by utilizing two cascading 16-bit Cyclic Redundancy Check(CRC) messages as mutual authentication factor between the tag and the reader. The analytical results show the proposed protocol possesses untraceability, authenticity and usability, it resists replaying attack and synchronization attack. But this scheme stores old secret key and new secret key in different tables. Sometimes it has to search from a table to another one and this takes much more time. Otherwise, there are some redundant data among two different tables and it is easy to result in data inconsistency, which will trigger DOS attack.

Sejin Oh et al.[12] proposed another authentication protocol for the C-1 G-2 RFID system. At first, they gave five assumptions: (1) the communication channel between the server and the reader is secure. (2) the communication channel between the reader and the tag is insecure. (3)the server and the tag can calculate hash function. (4) the tag, the reader and the server can generate a random number. (5) the tag and the server can operate a *CRC* coding and key-shift operation. Then they proposed an authentication protocol for the C-1 G-2 RFID system. They claimed that their proposed authentication protocol can prevent eavesdropping, location tracking, spoofing and replay attack. Their protocol uses hash function in the backend server and the tag. In fact, this does not conform to the C-1 G-2 RFID specification. Besides, the backend server uses hash function to calculate each entry in its database to find out a matched entry. This is a very heavy payload for the backend server. The generation of its secure key *C* uses a random number from the backend server and this random number is transferred to the tag by plaintext. This leads that it is easy to be intercepted by an adversary. Because the range of this random number is from 0 to 127 and it is very limited, so it is feasible to guess the shared secure key *k* and the tag's identifier *ID* if the adversary could intercept the certification key *C*.

Since the C-1 G-2 RFID specification was proposed in 2006, some authentication protocols which conform to this specification have been proposed. But as analyzed above, these protocols maybe have some flaws or they used some other functions which the C-1 G-2 RFID specification does not support. They cannot satisfy the requirements of the wide applications of the C-1 G-2 RFID tags.

4 A *CRC*-Based Lightweight Authentication Protocol for RFID

In the following parts, we use *CRC* function to encrypt the messages exchanged between readers and tags so as to assure the privacy and security of the RFID system. At the same time, a pseudorandom generator is used to assure the freshness of the exchanged messages. After each successful authentication the tag's identification information is updated so as to prevent being traced. As usual, we suppose the communication channel between the backend server and readers is secure. The used symbols during authenticating are listed in Table 1.

Supposed a tag is uniquely identified by its identifier *ID*. *Meta-ID* is the tag's pseudonym. The backend server shares the same *CRC* and pseudorandom function with the tag. All tags share the secret key k with the reader and the random numbers generated in tags or readers are encrypted by the secret key k . *Meta-ID* are stored in the tag. *old-ID*, *new-ID*, *old-Meta-ID* and *new-Meta-ID* are stored in the backend server. The purpose to set two sets of *ID* and *Meta-ID* for the backend server is to prevent the desynchronization attack or DOS attack. These parameters are 32 bits. *New-flag* is a flag bit in the backend server and it marks *new-Meta-ID* or *old-Meta-ID* is chosen to authenticate.

The tag and the backend server have two functions respectively. One is the *CRC* function $CRC()$, and another is the pseudorandom generator $PRNG()$. They are two 16-bit functions conforming to the C-1 G-2 RFID specification. Because some parameters to identify the tag(as described above) are 32 bits and in order to avoid collision from the function $CRC()$ we divide each 32-bit parameter into two parts by two other functions, which are $funh(x)$ and $funl(x)$, and each part is 16 bits so as to reduce the collision chance of 16-bit *CRC* function. These functions are also shown as table 1.

Table 1. The related notations for the *CRC*-based authentication protocol

Notation	Description
ID	the unique identifier of a tag
$Meta\text{-}ID$	the pseudonym of the tag
k	the 32-bit secret key shared by readers and tags
$CRC()$	the *CRC* function
R_r	the pseudorandom number generated by a reader
R_t	the pseudorandom number generated by a tag
$PRNG()$	the pseudorandom generator
$funh(x)$	the function to get the left half-part of x
$funl(x)$	the function to get the right half-part of x
$\oplus$	*XOR* operator
$\|\|$	concatenation operator

Under the initial state, *old-ID=new-ID=ID*, *old-Meta-ID=new-Meta-ID=Meta-ID*= $PRNG(funl(ID)) \| PRNG(funh(ID))$, *new-flag*=0.

The mutual authentication procedure between the reader/backend server and the tag is shown as Fig. 2 and this protocol is described as follows:

♦ Step1: reader→tag

The reader generates a query to the tag and it calls the pseudorandom function to get a random number $R_r = PRNG()$. Then it generates a message: $M1 = funh(k) \oplus R_r$, the reader sends query||*M1* to the tag.

♦ Step 2: tag→reader

The tag uses its k to get R_r from *M1* by XOR operation. The tag uses $PRNG()$ to generate another random number R_t and then it generates the messages: $M2 = CRC(funl(Meta\text{-}ID) \oplus R_t) \| CRC(funh(Meta\text{-}ID) \oplus R_r)$, $M3 = CRC(funl(ID) \oplus R_t \oplus R_r)$, $M4 = funl(k) \oplus R_t \oplus R_r$. The tag sends *M2*, *M3* and

M4 to the reader. After the reader receives *M2*, *M3* and *M4*, it abstracts R_t from *M4* by *XOR* operation. Then it sends *M2*, *M3*, R_r and R_t to the backend server.

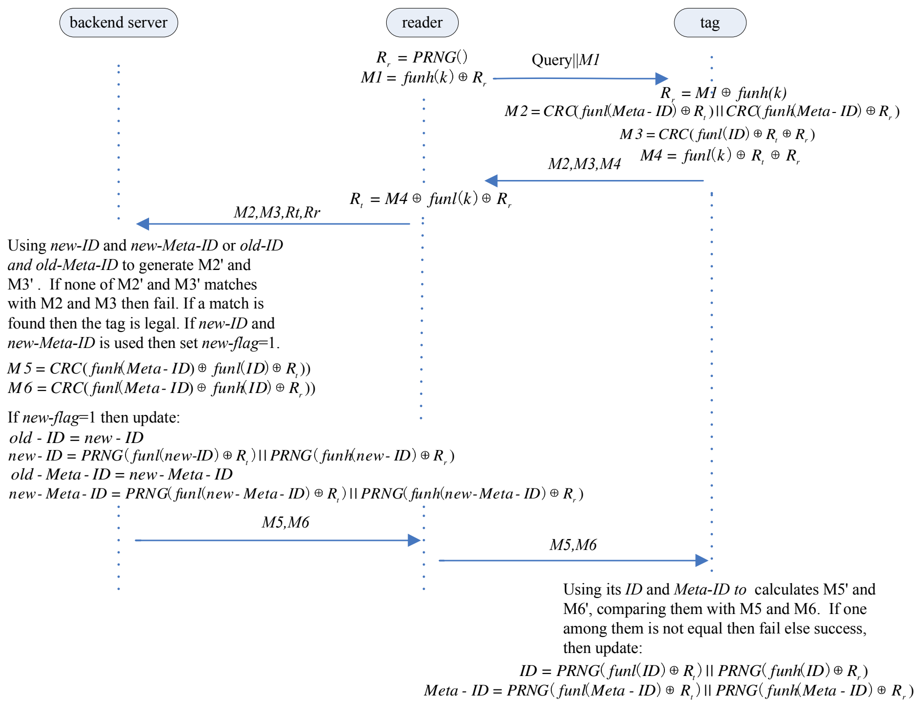

backend server
reader
tag
$R_r = PRNG()$
$M1 = funh(k) \oplus R_r$
Query||*M1*
$R_r = M1 \oplus funh(k)$
$M2 = CRC(funl(Meta\text{-}ID) \oplus R_t) \| CRC(funh(Meta\text{-}ID) \oplus R_r)$
$M3 = CRC(funl(ID) \oplus R_t \oplus R_r)$
$M4 = funl(k) \oplus R_t \oplus R_r$
M2,M3,M4
$R_t = M4 \oplus funl(k) \oplus R_r$
M2,M3,Rt,Rr
Using *new-ID* and *new-Meta-ID* or *old-ID and old-Meta-ID* to generate M2' and M3'. If none of M2' and M3' matches with M2 and M3 then fail. If a match is found then the tag is legal. If *new-ID* and *new-Meta-ID* is used then set *new-flag*=1.
$M5 = CRC(funh(Meta\text{-}ID) \oplus funl(ID) \oplus R_t))$
$M6 = CRC(funl(Meta\text{-}ID) \oplus funh(ID) \oplus R_r))$
If *new-flag*=1 then update:
$old\text{-}ID = new\text{-}ID$
$new\text{-}ID = PRNG(funl(new\text{-}ID) \oplus R_t) \| PRNG(funh(new\text{-}ID) \oplus R_r)$
$old\text{-}Meta\text{-}ID = new\text{-}Meta\text{-}ID$
$new\text{-}Meta\text{-}ID = PRNG(funl(new\text{-}Meta\text{-}ID) \oplus R_t) \| PRNG(funh(new\text{-}Meta\text{-}ID) \oplus R_r)$
M5,M6
M5,M6
Using its *ID* and *Meta-ID to* calculates M5' and M6', comparing them with M5 and M6. If one among them is not equal then fail else success, then update:
$ID = PRNG(funl(ID) \oplus R_t) \| PRNG(funh(ID) \oplus R_r)$
$Meta\text{-}ID = PRNG(funl(Meta\text{-}ID) \oplus R_t) \| PRNG(funh(Meta\text{-}ID) \oplus R_r)$

Fig. 2. The diagram of the *CRC*-based authentication protocol

♦ Step 3: backend server→reader

After receiving *M2*, *M3*, R_r and R_t, the backend server searches its database. It gets each record from its database. At first it uses *new-ID* and *new-Meta-ID* in this record to calculate $M3' = CRC(funl(ID) \oplus R_t \oplus R_r)$, $M2' = CRC(funl(Meta\text{-}ID) \oplus R_t) \| CRC(funh(Meta\text{-}ID) \oplus R_r)$, where *ID* and *Meta-ID* will be replaced with *new-ID* and *new-Meta-ID*. Then *M2'* and *M3'* are compared with *M2* and *M3*. If one among two pairs is not equal then *old-ID* and *old-Meta-ID* of this record are also used to calculate *M2'* and *M3'*, and to be compared with *M2* and *M3* again. After searching all records in the database they are all not equal yet and the back-end server notify the reader that the tag is illegal. Otherwise, a match is found, which *M2'* and *M3'* are equal to *M2* and *M3* respectively, then this states the tag is legal and the authentication to the tag is finished. If *M2'* and *M3'* which match successfully with *M2* and *M3* use *new-ID* and *new-Meta-ID*, *new-flag* is set one else zero. Then the backend server generates the messages $M5 = CRC(funh(Meta\text{-}ID) \oplus funl(ID) \oplus R_t))$ and $M6 = CRC(funl(Meta\text{-}ID) \oplus funh(ID) \oplus R_r))$, where *Meta-ID* and *ID* will be replaced with *old-Meta-ID* and *old-ID* or *new-Meta-ID* and *new-ID* dependable on the value of *new-flag*. If *new-flag* is one the backend server begins to update its key values as follows:

$old\text{-}ID=new\text{-}ID$

$new\text{-}ID = PRNG(funl(new\text{-}ID)\oplus R_t) \| PRNG(funh(new\text{-}ID)\oplus R_r)$

$old\text{-}Meta\text{-}ID=new\text{-}Meta\text{-}ID$

$new\text{-}Meta\text{-}ID = PRNG(funl(new\text{-}Meta\text{-}ID)\oplus R_t) \| PRNG(funh(new\text{-}Meta\text{-}ID)\oplus R_r)$

Finally the backend server sends *M5*, *M6* to the reader.

♦ Step 4: reader→tag

The reader receives *M5*, *M6* and sends them to the tag. The tag receives *M5* and *M6*. Then the tag calculates *M5'* and *M6'* by using *ID*, *Meta-ID*, R_t and R_r stored in it. If *M5'* is not equal to *M5* or *M6'* is not equal to *M6* the tag refuses to be accessed by the reader. Otherwise, the tag permits the access of the reader and this completes the authentication to the reader. Finally, the tag updates *ID* with $PRNG(funl(ID)\oplus R_t) \| PRNG(funh(ID)\oplus R_r)$, *Meta-ID* with $PRNG(funl(Meta\text{-}ID)\oplus R_t) \| PRNG(funh(Meta\text{-}ID)\oplus R_r)$.

5 The Analysis of the *CRC*-Based Authentication Protocol

During authenticating described above, the tag and the backend server only call *CRC* and pseudorandom generating operations respectively. These operations are supported by the tag conforming to the C-1 G-2 RFID specification. To enhance the authentication strength dual authentications between the reader and the tag are completed. Dual authentications mean that only if M2' and M3' are all equal to M2 and M3 the reader can finish the authentication to the tag, and only if M5' and M6' are all equal to M5 and M6 the tag can finish the authentication to the reader. If anyone among these messages is tampered the authentication between them will fail. At the same time some pseudorandom numbers generated by the tag or the reader is simply hidden by their shared secure key k.

Now we begin to analyze the security of the proposed authentication protocol as follows:

♦ Eavesdropping: During the authentication period, all messages exchanged between tags and readers are encrypted by *CRC* function and attackers do not know anything about the tag from their acquired data. Eavesdropping to the communication between tags and readers is invalid. The privacy of the RFID system is protected.

♦ Location tracing: One of the most serious privacy problems for the RFID system is that if an invariable value is exposed during each authentication, the privacy of the user's location may be encroached upon. To prevent this type of attack, a pseudorandom generator is used to assure each session between the tag and the reader is variable so as to make attackers not to know where their received data is from. Additionally, after each successful authentication *ID* and *Meta-ID* in the tag are updated. Next authentication will generate some new sessions different from the previous authentication so that an adversary cannot trace the location of a tag or its holder.

♦ Desynchronization attack or DOS attack: When the identification information of the tags stored in the backend server is not the same as in tags the mutual authentication between tags and readers will not continue any more. At this time we say that desynchronization attack or DOS attack occurs. Our proposed protocol avoids this attack effectively by providing one old set and one new set about *ID* and *Meta-ID* in the backend server. For our proposed protocol it is possible that M5 or M6 is intercepted by an attacker during authenticating and the tag does not received M5 or M6. Hence *ID* and *Meta-ID* in the backend server have been updated successfully but the corresponding information in the tag is not updated. The information not updated in the tag has been used as old value yet stored in the backend server. So at any time the identification information in the tag keeps the same as the old or new identification information in the backend server so as to ensure next authentication.

♦ Replay attack: This type of attack means to re-send data acquired by eavesdropping to compromise the RFID system. In order to prevent replay attack the message of each authentication between tags and readers should be different by randomizing and *CRC* operations. After each successful authentication some related information(e.g. *ID*, *Metal-ID*) is also updated. If an attacker re-sends its received message later this message has not any meanings because each new authentication generates a new random number and the different messages.

♦ Forward security: Because for each authentication the reader and the tag generate a pair of new random numbers that there does not exist any relationship with the last authentication. They use the different the tag's identifier from last authentication to generate sessions. Attackers cannot infer any useful information of the last authentication from the present received messages and they cannot guess the tag's or reader's past behaviors.

♦ Spoofing: The protocol ensures user anonymity and privacy by encrypting all exchanged messages between readers and tags by *CRC* function and pseudorandom generating function. An attacker cannot get the identity information of tags by eavesdropping, so it cannot impersonate a valid tag.

The comparison of the proposed protocol with other typical authentication protocols based on *CRC* is listed in Table 2.

Table 2. The comparison of the different authentication protocols

Protocol type	Eaves-dropping	Location tracing	DOS	Replay attack	Forward security	Spoofing
S. Karthikeyan et al.[6]	√	x	x	x	-	-
D.N. Duc et al.[7]	√	x	x	x	x	x
Chien et al.[5]	√	x	x	x	√	x
The proposed protocol	√	√	√	√	√	√

6 Conclusion

It is generally admitted that the security and privacy of the tag play an important role in determining the cost and performance of an RFID system. To solve this problem we have proposed a lightweight mutual authentication protocol based on *CRC* function and this protocol completes the mutual authentication between tags and readers. *CRC* function is also a one-way function like a hash function and it only ensures the integrity of the messages. For the authentication to low-cost RFID tags, especially the C-1 G-2 RFID tags, to use *CRC* function is a best compromising scheme. By analyzing it is obvious that our proposed protocol is superior to other similar protocols, our protocol assures forward security and it can prevent information leakage, location tracing, eavesdropping, DOS attack, replay attack, spoofing. The proposed protocol can make full use of the computing resources embedded in the tag and it takes some effective measurements to reduce the collision of *CRC* function. The proposed protocol completes the strong secure authentication between the tag and the reader by dual authentications and it is suitable for the low-cost RFID systems conforming to the EPCglobal C-1 G-2 RFID specification.

Acknowledgments. We are appreciated to Prof. Josef Pieprzyk and Dr. Christophe Doche of Macquarie University for their constructive suggestion to this paper. The relative work about this paper is supported by National Natural Science Foundation of China (No. 61272097), the Science and Technology Innovation Project of Shanghai Education Committee(No. 12ZZ182), and the Discipline Developing Foundation of Shanghai University of Engineering Science(No. XKCZ1212).

References

1. Nambiar, A.N.: RFID Technology: A Review of its Applications. In: Proceedings of the World Congress on Engineering and Computer Science, vol. II, pp. 1–7. IAENG, Hong Kong (2009)
2. Gao, L.J., Ma, M.D., Shu, Y.T., Wei, Y.H.: An Ultralightweight RFID Authentication Protocol with CRC and Permutation. Journal of Network and Computer Applications 10, 1–20 (2013)
3. Kang, S.Y., Lee, D.G., Lee, I.Y.: A Study on Secure RFID Mutual Authentication Scheme in Pervasive Computing Environment. Computer Communications 31, 4248–4254 (2008)
4. Weis, S.A., Sarma, S.E., Rivest, R.L., Engels, D.W.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 201–212. Springer, Heidelberg (2004)
5. Chien, H.Y., Chen, C.H.: Mutual Authentication Protocol for RFID Conforming to EPC Class 1 Generation 2 Standards. Computer Standards & Interfaces 29, 254–259 (2007)
6. Karthikeyan, S., Nesterenko, M.: RFID Security without Extensive Cryptography. In: Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, pp. 63–67. ACM, New York (2005)
7. Duc, D.N., Park, J., Lee, H., Kim, K.: Enhancing Security of Class 1 GEN-2 RFID Tag against Traceability and Cloning. In: The 2006 Symposium on Cryptography and Information Security, pp. 269–277. Springer, Heidelberg (2006)
8. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: Cryptanalysis of a Novel Authentication Protocol Conforming to EPC-C1G2 Standard. Computer Standards & Interfaces 31, 372–380 (2009)

9. Han, D., Kwon, D.: Vulnerability of an RFID Authentication Protocol Conforming to EPC Class 1 Generation 2 Standards. Computer Standards & Interfaces 31, 648–652 (2009)
10. Lo, N.W., Yeh, K.-H.: An Efficient Mutual Authentication Scheme for EPCglobal Class-1 Generation-2 RFID System. In: Denko, M.K., Shih, C.-s., Li, K.-C., Tsao, S.-L., Zeng, Q.-A., Park, S.H., Ko, Y.-B., Hung, S.-H., Park, J.-H. (eds.) EUC-WS 2007. LNCS, vol. 4809, pp. 43–56. Springer, Heidelberg (2007)
11. Hu, T., Wei, G.H.: Anonymous Bidirectional RFID Authentication Protocol Based on Low-cost Tags. Journal of Computer Applications 32(1), 111–114 (2012)
12. Oh, S., Shin, J., Jeong, C., Lee, J.: A Mutual Authentication Protocol in RFID Using CRC and Variable Certification Key. In: 2012 International Conference on Affective Computing and Intelligent Interaction. Lecture Notes in Information Technology, vol. 10, pp. 84–89. Springer, Heidelberg (2012)

Test Case Prioritization Based on Genetic Algorithm and Test-Points Coverage

Weixiang Zhang, Bo Wei, and Huisen Du

Beijing Institute of Tracking and Telecommunications Technology, Beijing, China
wxchung@msn.com

Abstract. By optimizing the execution order of test cases, test case prioritization techniques can effectively improve the efficiency of software testing. Test case prioritization is becoming a hot topic in software testing research. Combining genetic algorithm with test-points coverage, this paper obtains some meaningful research results in test case prioritization, especially for the functional testing. Firstly, presents two new test case prioritization evaluations APTC and its improvement APRC_C. As focused on test-points coverage, these evaluations are more suitable for black-box testing. Then, proposes a test case prioritization method based on genetic algorithm, whose representation, selection, crossover and mutation are designed for black-box testing. Finally, verifies the proposed method by experiments data. The experimental results show that the proposed method can achieve desired effect.

Keywords: Software Testing, Test Case Prioritization, Genetic Algorithm, Evaluation Function, Black-box Testing, Software Engineering.

1 Introduction

Software testing is one of the most important means to ensure software quality. Statistics show that software testing accounts for more than 50% of the total cost of software development in general [1]. With the increasing software complexity, software testing is becoming more and more difficult and expensive. How to select a few of test cases from huge available collection to test software effectively has become an outstanding problem.

Test case prioritization technology (TCP) is one of the most important research directions of software testing. According to the importance degree based on some specific criteria, TCP sorts the test cases and then executes them sequentially. By optimizing the execution order of test cases, TCP can effectively improve the efficiency of software testing.

A general description of TCP is as follow: for a given test case set T, the total permutation PT of T and a sort function $f: PT \to R$, to find $T' \in PT$, let $f(T') \geq f(T'')$, for $\forall T'' \in PT(T'' \neq T')$. A typical sort goal is to maximize the rate of early defect detection. Other sort goals can be considered such as code coverage, defect detection rate of high-risk, defect detection rate of modified code related and system reliability and so on.

The current researches focus on code-based test case prioritization techniques. Usually based on the coverage power to program (e.g. statement, branch or function) of every test case, set weights and apply the greedy method to guide test ordering [2]. Some researchers try using typical machine learning methods to improve the effect of the implementation of TCP technology, such as meta-heuristic search [3], Bayesian networks [4] and cluster analysis [5].

By simulating the process of biological evolution, genetic algorithm (GA) searches the optimal solution for the optimization problem. GA maintains a population of potential solutions; it randomly samples in the entire search space, and evaluates each sample in accordance with its fitness function. In the genetic algorithm, some operators such as selection, crossover and mutation are used, which constantly iterates (each iteration is equivalent to one cycle of biological evolution) to search for a global optimal solution, until the termination condition is met.

In the test data generation, the search space of GA is the input domain of the software and the optimal solution is some test data to meet for the specified testing purposes. GA is more often used in the structural testing, taking running path as optimization goal [6-9]. Researches on using GA in the functional testing are not very extensive, including a method based on Z language specification [10] and the method based on pre / post-conditions [11]. Because of the high cost of formalization, these methods are difficult for large systems.

Aiming at the functional testing, this paper proposed a new test case prioritization evaluation, named Average Percentage of Test-Points Coverage (APTC) and its improvement APRC_C. Then, using APTC or APTC_C as fitness function, we proposed a test case prioritization method based on genetic algorithm. At last, we verified our method by simulation experiments.

2 Evaluation of Test Case Prioritization

2.1 Some Existing Evaluation Function

The purpose of software testing is as much and as quickly as possible detection of defects in the software. Compared with software testing in random sequence, one of the biggest benefits of TCP is able to check out the errors faster. Rothermel pointed out that TCP can be evaluated by the relationship between the number of test cases executed and errors detected.

Rothermel proposed an evaluation named APFD (Average Percentage of Fault Detection). Elbaum [12] gave the formula of APFD.

Suppose test case set T contains n test cases and m defects can be detected. For given test cases sequence, TF_i represents the precedence of the first test case to detect the i^{th} defect in the sequence, so there is $APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n}$.

APFD ranges between 0 to 100%, the higher the value, the faster the defect detection.

Since people cannot predict defect detection information of test cases, Li Zheng et al [13] then proposed APBC (average percentage of block coverage), APDC (average percentage of decision coverage) and APSC (average percentage of statement coverage) that could evaluate the coverage power to program (block, decision, statement respectively) of the test cases sequence. Formulas of these evaluations are similar to APFD, e.g. $APBC = 1 - \frac{TB_1 + TB_2 + \cdots + TB_m}{nm} + \frac{1}{2n}$, where TB_i represents the precedence of the first test case to cover the i^{th} block.

As the coverage information can be obtained through coverage analysis tools without test cases execution, so it is feasible to use APBC, APDC and APSC before software testing is over. However, these evaluations are clearly more suitable for structural testing or white-box testing.

Therefore, we propose a new test case prioritization evaluation based on test-points coverage.

2.2 Average Percentage of Test-Points Coverage (APTC)

In black-box testing, testers design test cases based on software specifications. Firstly, transform the software requirements into the software test requirements using of requirements analysis; secondly, decompose the test requirements into a lots of test points; then, design test cases aimed at test points respectively and then form a collection of test cases. Each test case may correspond to one or more test points.

For a test cases collection $\Phi = \{T_1, T_2, \cdots, T_m\}$, define APTC(average percentage of test-point coverage) as follows:

$$APTC = 1 - \frac{TT_1 + TT_2 + \cdots + TT_m}{nm} + \frac{1}{2n} \tag{1}$$

Wherein, n denotes test cases count, m denotes test-points count, TT_i represents the precedence of the first test case to cover the i^{th} test-points. APTC ranges between 0 to 100%, the higher the value, the faster the test-points coverage.

Taking into account the different importance level of each test-point and the different cost of each test case, adjust the evaluation object to the importance value of test-points covered by unit test case cost. Its formulation is expressed as follows:

$$APTC\_C = \frac{\sum_{i=1}^{m}\left(f_i \times \left(\sum_{j=TT_i}^{n} t_j - \frac{1}{2} t_{TT_i} \right) \right)}{\sum_{j=1}^{n} t_j \times \sum_{i=1}^{m} f_i} \tag{2}$$

Wherein, f_i represents the value of the importance of the i^{th} test-point, t_j represents the cost of the j^{th} test case, and the other variables is consistent with formula (1). When the importance of all test-points is the same and the costs of all test cases is equal, equation (2) reduces to equation (1). That is, APTC_C degenerates into APTC.

Consider the examples shown in Table 1, in which there are 5 test-points and 10 test cases. For test case sequences A-B-C-D-E and E-D-C-B-A, the APTC values is 50% and 64% respectively, so the effect of the latter is better than the former. Figure 1 shows the relationship between the rate of executed test case and the rate of covered test-points in different test case sequences. APTC is equal to the ratio of the area of the shaded portion under the line to whole area.

Table 1. An example of the relationship between test cases and test-points

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
A	X				X					
B	X				X	X	X			
C	X	X	X	X	X	X	X			
D					X					
E								X	X	X

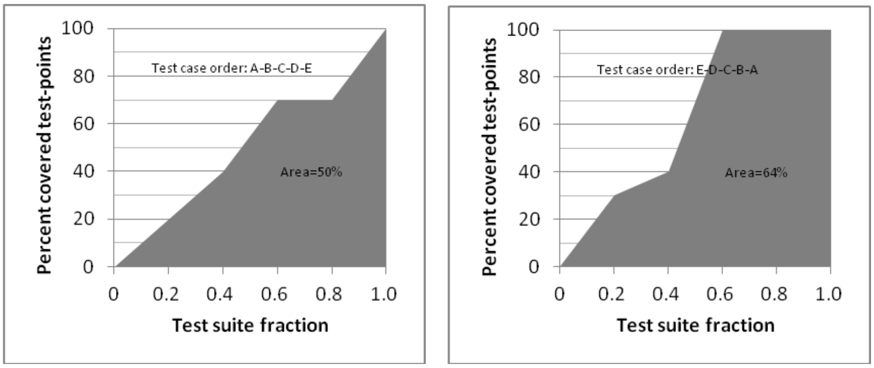


Fig. 1. APTC of different test case sequences

Supposed that the cost of test case B is two times of the other test cases, and the importance value of test-points covered by test case B (that is, test-point 1, 5, 6, 7) is two times of the other test-points. According to the formula, it is easy to calculate that APTC_C of A-B-C-D-E and E-D-C-B-A are 56% and 68%.

After replacing the horizontal and vertical coordinates by "test suite fraction with cost" and "percent covered test-points with importance", we can draw the diagram of APTC_C in a similar way to APTC, as shown in Figure 2.

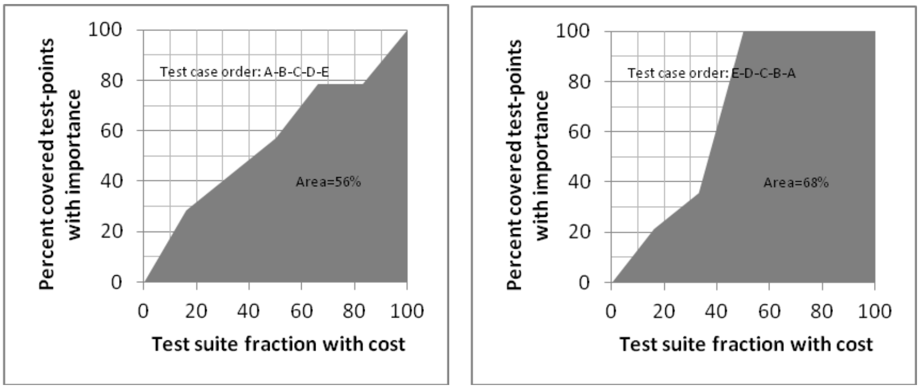


Fig. 2. APTC_C of different test case sequences

In the case of large differences in test-points' importance or/and test cases' costs, APTC_C is more appropriate than APTC.

3 Test Case Prioritization Based on Genetic Algorithm

3.1 Process of Genetic Algorithm

In genetic algorithm (GA), each effective solution to the problem is called a "chromosome", with respect to each individual of population. A chromosome is a coded string using a specific encoding method, and each unit of the coded string is called a "gene". By comparing the fitness values, GA distinguishes the pros and cons of chromosomes. The chromosome with larger fitness value is more outstanding.

In GA, fitness function is used to calculate the fitness value of corresponding chromosome; selection is used to choose some individual in accordance with certain rules, and form the parent population; crossover is used to interchange part of genes of two individuals to generate their offspring chromosomes; mutation is used to change a few genes of selected chromosome to get a new one.

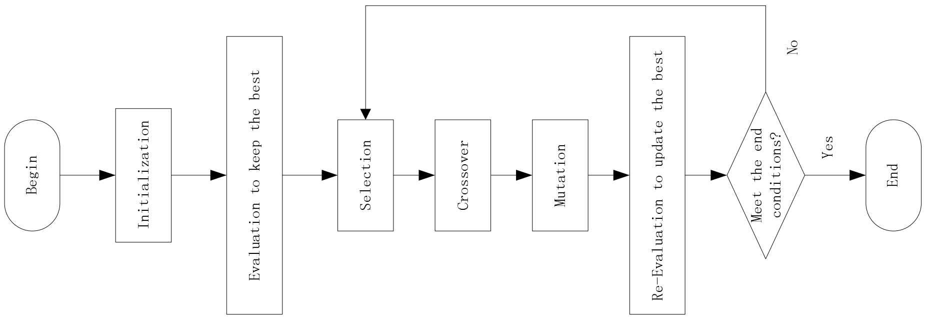


Fig. 3. Flowchart of GA

The main steps of GA include:

STEP1. To Initialize a population with N chromosomes, get the genes of every chromosome in random manner and keep them inside the range of the problem definition. Denote the count of generation $Generation$ and let $Generation = 0$.

STEP2. To evaluate each chromosome using fitness function, calculate the fitness value of every chromosome, save the best one whose fitness is largest and name it $Best$.

STEP3. To do selection using of the manner such as Roulette wheel, generate the population with N selected chromosomes.

STEP4. To do crossover in accordance with the probability p_c. Each couple of selected chromosomes interchange some genes to generate their two offspring and replace themselves; other chromosomes retain in the population.

STEP5. To do mutation in accordance with the probability p_m. Some new chromosomes are generated separately through altering a few genes of corresponding selected chromosome; other non-selected chromosomes retain in the population.

STEP6. Re-evaluate each chromosome using the fitness function. If the largest fitness value in the new population is better than $Best$'s, replace $Best$.

STEP7. Let $Generation{+}{+}$. If $Generation$ exceeds the specified maximum generation or $Best$ achieves the specified error requirement, end the algorithm; otherwise, goto STEP3.

3.2 Design of Representation

In TCP, each chromosome is a test case sequence. What we concern is the order of test cases instead of their specific composition. So, we assign a unique natural number for every test case, and then each individual can be encoded as an ordered sequence of the numbers. The scale of initial population affects to the search capability and operational efficiency of GA, so it usually range from 20 to 150.

For an example, the representation of an individual $T_1 - T_2 - T_3 \cdots - T_m$ of test cases suite $\Phi = \{T_1, T_2, \cdots, T_m\}$ is shown as in Figure 4.

T1	T2	T3	···	Tm

Fig. 4. Design of representation

3.3 Design of Fitness Function

Fitness function is used to calculate the fitness value of each chromosome and to guide the search direction of GA. So, it is the key part of genetic algorithm implementation. Generally, the fitness value is between 0 and 1. The individuals with larger value are more excellent, and have greater probability to evolve to the next generation.

In general, for a test case, the more the count of test points to cover, the larger the probability to find defects. Therefore, taking consider of improving the coverage of test cases to test-points as evolutionary goal, APTC is good to be used as the fitness function. Furthermore, if the goal is designed to improve the importance degree of covered test-points in unit test cost, APTC_C is more suitable for fitness evaluation.

3.4 Design of Selection

Use roulette wheel selection. For example, for a population with k individuals, denotes $fitness_i$ for fitness of i^{th} individual, the roulette wheel selection include 5 steps: first, calculate the fitness percent $fitness_i / \sum fitness_i$ which show the capability of each individual to yield offspring; second, sort the individuals by descending order of their fitness percent; third, for each individual, sum up all the fitness percent of individuals that are in ahead of it; then, select the first individual whose summed fitness is greater than the random number $r_s \in [0,1]$; lastly, loop above steps until enough individuals is born. As can be seen, the individuals with greater fitness will have larger probability to be selected to produce the next generation, which is consistent with the principles of evolution.

3.5 Design of Crossover

Suppose Q_1 and Q_2 are the selected individuals to do crossover, D_1 and D_2 are the individuals after crossover. Set crossover probability p_c and get a random number $r_c \in [0,1]$, obtain D_1 and D_2 when r_c is less than p_c in a manner as follows. For a random crossover point $k\left(1 \le k \le m\right)$, D_1 consists of two parts: the first part is the first k test cases of Q_1; the second part is from Q_2 excluding the first k test cases of Q_1. Similarly, D_2 consists of two parts: the first part is the first $\left(m-k\right)$ test cases of Q_2, the second part is from Q_1 excluding the first $\left(m-k\right)$ test cases of Q_2. Figure 5 shows an example. The value of p_c is generally between 0.4 and 0.99.

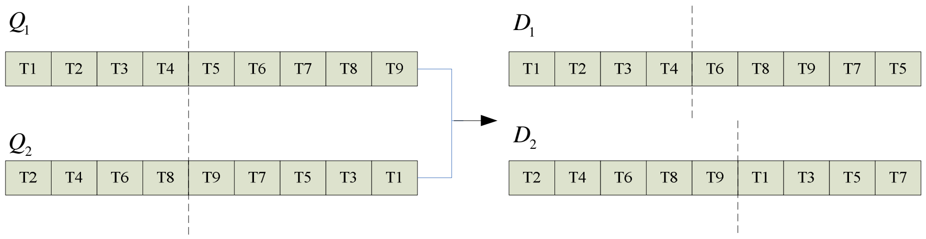

Q_1
T1 T2 T3 T4 T5 T6 T7 T8 T9
Q_2
T2 T4 T6 T8 T9 T7 T5 T3 T1
D_1
T1 T2 T3 T4 T6 T8 T9 T7 T5
D_2
T2 T4 T6 T8 T9 T1 T3 T5 T7

Fig. 5. Design of crossover

3.6 Design of Mutation

Mutation change some genes of the selected chromosome to generate a new individual. Set mutation probability p_m and get a random number $r_m \in [0,1]$, obtain the new individual when r_m is less than p_m in a manner as follows: according to the importance of test-points covered by test cases, select two test cases of the parent individual using of the manner as roulette wheel selection; then, exchange the positions of the two selected test cases to form a new individual. Figure 6 shows an example. The value of p_m is generally between 0.001 and 0.1.

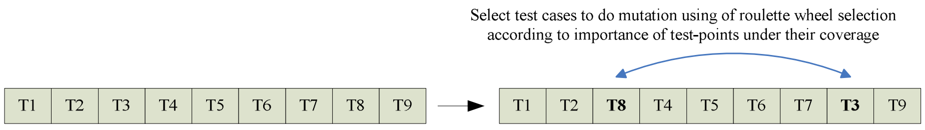

Select test cases to do mutation using of roulette wheel selection according to importance of test-points under their coverage
T1 T2 T3 T4 T5 T6 T7 T8 T9
T1 T2 T8 T4 T5 T6 T7 T3 T9

Fig. 6. Design of mutation

4 Simulation Experiment

We use triangle classification program to illustrate the effect of the proposed method. Triangle classification program analyzes the values and their relationship of the three input variables (x, y, z) to determine the type of the triangle composed by them. There are total 7 test-points of triangle classification program, as shown in Table 2. We design total 6 test cases by black-box testing method. The relationship between test cases and test-points is shown in Table 3.

Table 2. Test-points of triangle classification program

No.	Name	Description	Importance
1	IsNumError	Judge if (x, y, z) is out of the valid range or not, i.e., check whether each variable is less than 0 or greater than the maximum value.	1
2	IsTriangleError	Judge if (x, y, z) is able to form triangle or not, i.e., check whether the sum of any two sides is large than the other side.	1
3	IsTriangle	Judge if (x, y, z) is suitable for triangle sides or not, i.e., check whether each variable is within the valid range and the sum of any two sides is large than the other side.	1
4	IsScalTriangle	Judge if the triangle is inequilateral or not in the condition that (x, y, z) can form triangle, i.e., check whether x ≠ y ≠ z.	2
5	IsRightTriangle	Judge if the triangle belongs to right triangle or not in the condition that (x, y, z) can form triangle, i.e., check whether x^2+y^2=z^2.	2
6	IsIsosTriangle	Judge if the triangle belongs to isosceles triangle or not in the condition that (x, y, z) can form triangle, i.e., check whether x=y、y=z or x=z.	2
7	IsEquiTriangle	Judge if the triangle belongs to equilateral triangle or not in the condition that (x, y, z) can form isosceles triangle, i.e., check whether x=y =z.	3

Table 3. Relationship between test cases and test-points of triangle classification program

No.	Test case name	Test case cost	1	2	3	4	5	6	7
A	NumErrorTest	1	X						
B	TriangleErrorTest	1		X					
C	ScalTriangleTest	2			X	X			
D	RightTriangleTest	2			X		X		
E	IsosTriangleTest	2			X			X	
F	EquiTriangleTest	3						X	X

Set the values of GA parameters as shown in Table 4. The best three results is obtained by the proposed method using of APTC and APTC_C separately, as shown in Table 5 and Table 6.

Table 4. Values of GA parameters in triangle classification program

No.	Name	Value
p_c	Probability of crossover	0.85
p_m	Probability of mutation	0.05
$G_{\max}$	Maximum count of iterations	50
N	Scale of population	150

Table 5. Partial results of APTC althgram

No.	Sequence of test cases	APTC
1	C-F-D-E-A-B/D-C-E-F-A-B/F-E-D-C-A-B	0.4881
2	A-C-D-F-E-B/C-A-D-E-F-B/F-D-C-A-E-B	0. 4643
3	A-C-D-B-F-E/C-A-D-B-F-E/D-B-F-A-C-E	0. 4405

Table 6. Partial results of APTC_C althgram

No.	Sequence of test cases	APTC_C
1	D-C-E-A-B-F/D-C-E-B-A-F	0. 8030
2	C-D-E-A-B-F/C-D-E-B-A-F	0. 7955
3	D-C-A-E-B-F/D-C-B-E-A-F	0. 7879

5 Conclusion

The use of genetic algorithms in test case prioritization, can effectively reduce the blindness in test cases executed order and so improve the efficiency of software testing.

This paper proposed a new test case prioritization evaluation APRC and its improvement APRC_C for functional testing. As focused on test-points coverage, these evaluations are more suitable for black-box testing.

In addition, this paper presented an automated test case prioritization method using of genetic algorithms which adopted APTC or APTC_C as fitness function. The designs of representation, selection, crossover and mutation in GA are aimed at black-box testing. We gave the specific steps of the method and validated it by experimental data.

The experimental results show that the proposed method can achieve expected results. It provides an effective technical approach to the test case prioritization problem. In the future, we will do further research in test-points automatically conversation and applications of GA in the automated generation of black-box test cases.

References

1. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press, Cambridge (2008)
2. Chen, X., Chen, J.H., Ju, X.L., Gu, Q.: Survey of test case prioritization techniques for regression testing. Journal of Software 24(8), 1695–1712 (2013) (in Chinese)
3. Li, Z., Harman, M., Hierons, R.M.: Search algorithms for regression test case prioritization. IEEE Trans. on Software Engineering 33(4), 225–237 (2007)
4. Mirarab, S., Tahvildari, L.: A prioritization approach for software test cases based on bayesian networks. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 276–290. Springer, Heidelberg (2007)
5. Yoo, S., Harman, M., Tonella, P., Susi, A.: Clustering test cases to achieve effective and scalable prioritization incorporating expert knowledge. In: Proc. of the Int'l Symp. on Software Testing and Analysis, pp. 201–212. ACM Press (2009)
6. Jones, J.A., Harrold, M.J.: Test-suite reduction and prioritization for modified condition/decision coverage. IEEE Trans. Software Eng. 29(3), 195–209 (2003)
7. Quan, J., Lu, L.: Research test case suite minimization based on genetic algorithm. Computer Engineering and Applications 45(19), 58–61 (2009)
8. Lin, J.C., Yeh, P.L.: Using genetic algorithms for test case generation in path testing. In: Proc. of the Asian Test Symposium, pp. 241–246 (2000)
9. Jones, B.F., Sthamer, H.H., Eyres, D.E.: Automatic structural testing using genetic algorithms. Software Engineering Journal 11(5), 299–306 (1996)
10. Baresel, A., Sthamer, H., Schmidt, M.: Fitness function design to improve evolutionary structural testing. In: Genetic and Evolutionary Computation Conference, pp. 1329–1336. IEEE Press, New York (2002)
11. Wegener, J., Buhler, O.: Evaluation of different fitness functions for the evolutionary testing of an automatic parking system. In: The Genetic and Evolutionary Computation Conference, pp. 1400–1412. Seattle, Washington (2002)

12. Elbaum, S., Malishevsky, A., Rothermel, G.: Prioritizing test cases for regression testing. In: Proc. of the Int'l Symp. on Software Testing and Analysis, pp. 102–112. ACM Press (2000)
13. Elbaum, S., Malishevsky, A., Rothermel, G.: Incorporating varying test costs and fault severities into test case prioritization. In: Proc. of the Int'l Conf. on Software Engineering, pp. 329–338. IEEE Press, New York (2001)

SAEP: Simulated Annealing Based Ensemble Projecting Method for Solving Conditional Nonlinear Optimal Perturbation

Shicheng Wen[1], Shijin Yuan[1,*], Bin Mu[1], Hongyu Li[1], and Lei Chen[2]

[1] School of Software Engineering, Tongji Univeristy, Shanghai
[2] LASG, Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing
{2013_wenshicheng,yuanshijin,binmu,hyli}@tongji.edu.cn,
qqydss@163.com

Abstract. Conditional nonlinear optimal perturbation (CNOP) is an initial perturbation evolving into the largest nonlinear evolution at the prediction time. It has played an important role in predictability and sensitivity studies of nonlinear numerical models. The popular solution for CNOP is the spectral projecting gradient algorithm based on the adjoint model. However, many modern numerical models have no adjoint models, and new implementations of adjoint models are quite a huge work. It thus leads to a limitation of CNOP applications. To alleviate the limitation, we propose a simulated annealing based ensemble projecting method free of the adjoint models to solve CNOP. To demonstrate the validity, we take the CNOP of the Zebiak-Cane model as a case to show the application of the proposed method. Also, a comparison is made between the adjoint-based method and the proposed method. Experimental results show that the proposed method can be treated as an approximate solution to CNOP.

Keywords: Simulated annealing, PCA, CNOP, ZC model.

1 Introduction

Predictability and sensitivity of atmospheric or oceanic motions are the critical issue in the numerical weather and climate prediction. Lorenz introduced the concepts of linear singular vector (LSV) and linear singular value (LSVA) [1]. This linear approach has been widely applied to computing the fastest-growing perturbations [2], and it has been also extended to explore error growth, predictability and targeted areas in adaptive observations [3]. However, the theories of LSV and LSVA are linear and limited in describing nonlinear evolution of finite-amplitude initial perturbation.

As one attempt for studying the predictability of a nonlinear system, the conditional nonlinear optimal perturbation (CNOP) was proposed by Mu and Duan for determining the initial perturbation with the largest nonlinear evolution [4]. Subsequently,

[*] Corresponding author.

CNOP is applied to studying the optimal precursor of El Nino-Southern Oscillation (ENSO) and the effect of nonlinearity on error growth for ENSO [5-6]; conduct the ensemble forecast [7]; identify the sensitive areas in targeting for the typhoon prediction [8-9]; and explore the nonlinear sensitivity and stability of the ocean's THC to finite amplitude perturbations [10].

The common method for solving CNOP is the spectral projecting gradient (SPG2) [11]. The algorithm requires two inputs: the objective function and corresponding gradient. And the gradient is always computed by the adjoint models. Unfortunately, many modern numerical models have no corresponding adjoint models, and it is quite a huge work to implement the new. It is of necessity to explore new approach for CNOP. On the one hand, some other researchers proposed the ensemble method for solving CNOP [12-13] still based on SPG2. Although Chen [13] computes the gradient by definition to avoid the adjoint model, the definition based method is limited especially when the model is complex. On the other hand, researchers start considering those optimization algorithms without gradient. Undoubtedly, intelligent algorithms are a good option. The genetic algorithm (GA) was introduced for solving CNOP in the case of models with the "on-off" switches [14-15]. Ye et al. proposed an improved particle swarm optimization to solve CNOP, and further compared it with the adjoint based approach [16]. However, these related works just took some simple and ideal models, such as the Lorenz model, as the object. It is due to that the complex models always accompany with high dimensional data. The problem of high dimensionality is just a bottleneck of the intelligent algorithms. In this paper, we propose utilizing the simulated annealing algorithm [17] based on principal component analysis (PCA) [18] to solve CNOP of practical models, and perform the case study of the Zebiak-Cane model which is famous for the successful forecast of the ENSO event.

The rest of the paper is organized as follows: Section 2 introduces some related knowledge. In section 3, we come up with the simulated annealing based ensemble projecting method. Section 4 reports the experimental results. This paper ends with the conclusion and future work in Section 5.

2 CNOP

CNOP is the perturbation $x_{0\delta}^{*}$ which makes the target function $J(x_0)$ achieve the maximum with a condition of $\|x_0\| \le \delta$, i.e.

$$\begin{aligned} &J(x_{0\delta}) = \max \|J_{TE}\|^2, \qquad s.t. \quad \|x_0\|_A \le \delta \\ &J_{TE} = M_{t_0 \to t}(\bar{x}_0 + x_0) - M_{t_0 \to t}(\bar{x}_0), \end{aligned} \tag{1}$$

where $M_{t_0 \to t}$ denotes the propagator of a nonlinear model from the initial time t_0 to the prediction time t, J_{TE} the nonlinear evolution of the initial perturbation, x_0 the initial perturbation, and δ is the magnitude of uncertainty. Here, both $\|\cdot\|_A$ and

$\|\cdot\|$ denote the L^2 norm. For the convenience of optimization, Eq. (1) can also be converted to a minimum problem as follows:

$$J(x_{0\delta}) = \min(-\|J_{TE}\|^2), \qquad s.t. \quad \|x_0\| \le \delta \tag{2}$$

Therefore, the gradient of J can be further obtained as:

$$\nabla J = -2 * H^T * J_{TE} \tag{3}$$

where H is the tangent matrix of the nonlinear model $M_{t_0 \to t}$, and its transpose H^T is so-called Jacobian matrix of J_{TE}. Here, the matrix, H^T is computed by corresponding adjoint models. However, it is difficult for those modern numerical models without the adjoint models to compute CNOP.

3 Simulated Annealing Based Ensemble Projecting Method

In previous studies [19-20], it is concluded that a driven dissipative system can get into a steady state that consists of attractors with low dimension after a long evolution. Any other state, that is, can be projected onto the space expanded by the attractors. Based on this conclusion, we make an assumption that CNOP can also be projected on the space. Thus we need to compute a group of orthogonal basis in the space of the attractors first. Then, the optimization algorithm is utilized to obtain CNOP in the low-dimensional space.

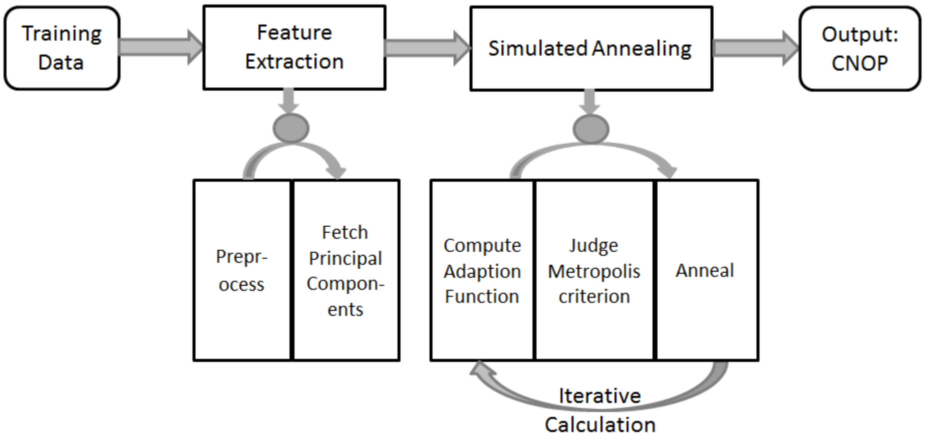


Fig. 1. The framework of SAEP

As is shown in Fig. 2, the simulated annealing based ensemble projecting method (for short, SAEP) mainly consists of two steps: the feature extraction and simulated annealing. Specifically, the training data need to be preprocessed first, and then the principal components can be obtained by feature exaction methods. In the simulated annealing step, first of all, initialize the state and generate a candidate; then utilize the Metropolis acceptance criterion to determine whether the candidate should be accepted; finally, update the temperature and go into the next iteration.

3.1 Feature Extraction

As discussed above, we should run the model for a long time from an initial state to obtain a data set, $x_1, x_2 \ldots x_n, x_i \in R^l \ (i = 1,2 \ldots n)$. We treat it as the training data, $X \in R^{l \times n}$. Before further processing, we should perform the preprocessing including dimensionless and centering. The dimensionless process aims to get rid of the influence of dimension to better describe real physical laws.

$$x_i = x_i / a \ (i = 1, \ldots n), \tag{4}$$

where a is a positive coefficient. And the centering guarantee the data mean equal to zero:

$$x_i = x_i - \overline{x} \ (i = 1, \ldots n), \ \overline{x} = \frac{1}{n} \sum x_i \ . \tag{5}$$

Model variables always locate in a high dimensional space while the attractors are in a low one. Thus we adopt feature exaction methods to find the space of attractors. As one popular method of dimensional reduction, PCA can determine the statistical principal direction of a training dataset. That is, the extracted features have statistical meaning which is similar with the character of attractors. Specifically, the principal components can be acquired by the eigen-decomposition:

$$XX^T P = P \Sigma, \tag{6}$$

where Σ is a diagonal matrix whose diagonal entries correspond to the eigenvalues of XX^T. Since XX^T is a positive definite matrix, the columns of the obtained eigenvectors P are mutually orthogonal. Then take the $k \ (k \ll n)$ eigenvectors corresponding to the top k biggest eigenvalues. It will be further discussed that how many eigenvectors should be taken later.

3.2 Simulated Annealing

After obtaining the feature space, it is required to project the perturbation onto the space. That is, the perturbation x_0 can be represented by a linear combination of the principal components. Replacing x_0, Eq. (2) is converted into:

$$\begin{gathered} J(\omega_{0\delta}) = \min(-\left\| M_{t_0 \to t}(\overline{x}_0 + P \cdot \omega) - M_{t_0 \to t}(\overline{x}_0) \right\|) \\ s.t. \quad \left\| P \cdot \omega \right\| \le \delta \end{gathered} \ . \tag{7}$$

Actually, the constraint, $\|P \cdot \omega\| \leq \delta$ can be further simplified,

$$\begin{aligned} \|P \cdot \omega\| &= (P \cdot \omega)^T (P \cdot \omega) \\ &= \omega^T P^T P \omega \\ &= \omega^T \omega \ (P^T P = E) \\ &= \|w\| \leq \delta. \end{aligned} \tag{8}$$

Therefore, the final optimal objective becomes:

$$\begin{aligned} J(\omega_{0\delta}) &= \min(-\|M_{t_0 \to t}(\overline{x}_0 + P \cdot \omega) - M_{t_0 \to t}(\overline{x}_0)\|) \\ s.t. & \quad \|\omega\| \leq \delta. \end{aligned} \tag{9}$$

Here, the argument for the objective function has been changed into the coefficient ω rather than x_0. Because the coefficient ω locate in a space with the dimension $k\ (k \ll n)$, it can be obtained by utilizing the simulated annealing algorithm on the space.

It is worth noting that we do not transform Eq. (9) that is an optimal problem with conditions into a Lagragian function. Instead, we utilize the constraint condition to guarantee all points locating in a desired super-sphere zone. When any point gets out of the zone, it will be pulled back onto the sphere. The specific process of SAEP is described as follows:

Generate an Initial Point Randomly and Initialize the States.
The initial point ω need to be constrained in the super-sphere. If the point goes out of the boundary, it must be pulled back through the following rule:

$$w = \begin{cases} w & \|w\| <= \delta \\ \dfrac{\delta}{\|w\|} \times w & \|w\| > \delta \end{cases} \quad i = 1, \cdots, m. \tag{10}$$

Compute the Adaption Value.
In the case of CNOP, Eq. (7) gives the adaption function. We just run the nonlinear model to get the corresponding adaption value.

Metropolis Criterion.
The candidate solution w^* has to enter into the judge of the Metropolis criterion [21], which models how a thermodynamic system moves from the current state w to a new state with minimized energy contents. The candidate solution, w^*, is accepted as the current solution based on the acceptance probability,

Algorithm 1. SAEP algorithm

```
Input: training set $X$
Initialization:
1: Set the parameters $k$ , $T$ , $L$ , $max\_iter$ and $\alpha$ ;
Feature extraction:
2: Perform the dimensionless and centering to $X$ ;
3: Make the eigen-decomposition of $XX^T$ ;
4: Take the top $k$ eigenvectors as principal components;
Simulated Annealing:
5: Randomly generate an initial solution $w$ ;
6: $project(w,\delta)$ ; ▷project the point back onto the boundary
7: while the termination is not satisfied do
8:   for $i \leftarrow 1:L$   ▷ $L$ is the Markov length do
9:     Generate a candidate solution $w*$ ;
10:    $project(w*,\delta)$ ;
11:    Calculate $\Delta_{w,w*} \leftarrow adaption\_func(w*) - adaption\_func(w)$ ;
12:    if $\Delta_{w,w*} \le 0$ , then $w \leftarrow w*$ ;
13:    else $w \leftarrow w*$ with probability in Eq. (9)
14:    end if
15:        If $adaption\_func(w*) < adaption\_func(w_{opt})$ then
16:      Update the current optimum, $w_{opt}$ ;
17:    End if
18:  end do
19:  Update T;  ▷update the temperature
20: end while
Output: the optimal solution $w_{o\delta}$
```

$$P_{Accept} = \begin{cases} \exp[-(f(w*) - f(w))/t_k] & if\ f(w*) > f(w) \\ 1 & if\ f(w*) > f(w). \end{cases} \tag{11}$$

Moreover, if the candidate solution is better than the current optimum, then update the current optimum by the candidate.

Annealing.
When completing the calculation of all states in the Markov chain with the length L, one should adjust the temperature, called the annealing process. The rule of adjusting temperature is as follows:

$$T_{k+1} = \alpha T_k . \tag{12}$$

α means the decay rate, and $T_i(i = 1, 2, 3\cdots)$ denotes the temperature in the i-th iteration.

Judge Whether the Termination Condition Is Satisfied.
The common termination condition in SA is that the iteration achieves convergence. That is, there is not a big change between the previous solution and the current one. However, it is just the ideal situation. In the practical applications, it is a commend way to set an upper bound of iteration, *max_iter*. That is, we don't need to make the program converge every time. When the algorithm converges or the maximum iteration reaches, the program ends up.

The pseudo-code of the complete SAEP algorithm for solving CNOP is described in Algorithm 1.

4 Experiments

Our experiments run on a HP Z800 workstation with an Intel Xeon E5645 2.4 GHz CPU and 8 GB of memory. To show the validity of the proposed method, we apply it to the CNOP of the Zebiak-Cane (ZC) model [22]. In the case, x_0 in Eq. (2) is a perturbation vector composed of the sea surface temperature anomalies (SSTA) and thermocline depth anomalies (THA). Also, we perform a comparison between the adjoint-based method [23] (for short, the XD method). The experimental results show that the SAEP method can be treated as an approximate solution to CNOP, and more importantly it is free of the adjoint models.

4.1 ZC Model

The ZC model successfully forecasted the El Nino event of 1986-1987. Since then, it has been widely applied to the research of the predictability and dynamics of ENSO [24]. It is a mesoscale air-sea coupled model for the Tropical Pacific, and consists of three sub-models: the atmospheric model, ocean model and coupled model.

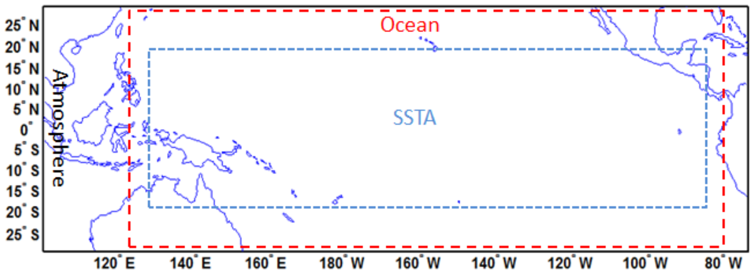


Fig. 2. The region of the ZC coupled model

Atmospheric Model.
The horizontal structure of the ZC atmospheric model can be denoted with the linear shallow-water equations of steady state in the equatorial β plane. Considering moisture convergence feedback, each step of the iterative procedure in the ZC atmospheric model is dependent on the divergence of the previous step. The ZC atmospheric model covers the region of $101.25^{\circ}E - 73.125^{\circ}W$, $29^{\circ}S - 29^{\circ}N$ with the resolution of $5.25^{\circ} \times 2^{\circ}$.

Ocean Model.
The ZC ocean model treats the mixing layer above the thermocline as being fluid, while the layer below as being stationary. Due to the time evolution of the sea surface temperature (SST) determined by horizontal advection, upwelling and heat loss to the atmosphere, the mixing layer is thus divided into two layers, the surface and subsurface layers. The ZC ocean model covers $124^{\circ}E - 80^{\circ}W$, $28.75^{\circ}S - 28.75^{\circ}N$ with the resolution $2^{\circ} \times 0.5^{\circ}$.

Coupled Model.
The ZC ocean model is driven by the anomaly of the wind stress forcing of sea surface which is obtained from the atmospheric wind field anomaly and the climate mean state. In the coupled model, the ZC ocean model can generate the sea surface temperature anomaly in the Tropical Pacific. It covers the region $129.375^{\circ}E - 84.375^{\circ}W$, $19^{\circ}S - 19^{\circ}N$ with the resolution of $5.25^{\circ} \times 2^{\circ}$.

The whole region of the ZC coupled model is shown is Fig. 2. The inner rectangle in the blue short dash is the integration region of SSTA in the coupled model. The middle rectangle in the red long denotes the region of the ocean model. The outer rectangle of the solid line represents the region of the atmospheric model.

4.2 Parameter Selection

In Section 3.2, we have talked about taking the k eigenvectors corresponding to the k top eigenvalues. How to set the value of k is unsolved nonetheless. To achieve this purpose, the following two aspects should be taken into consideration at least. The k eigenvectors should keep as much as energy, but k also should be not too big due to the problem of convergence and running time. Therefore, we propose utilizing the accumulative eigenvalues ratio to depict the conserved energy, and its gradient to demonstrate the increment rate of energy. The accumulative eigenvalue ratio is defined as:

$$R_{acc}^{(k)} = \frac{\sum_{i=1}^{k} \lambda_i}{\Omega}, \text{ where } \Omega = \sum_{i=1}^{n} \lambda_i . \tag{13}$$

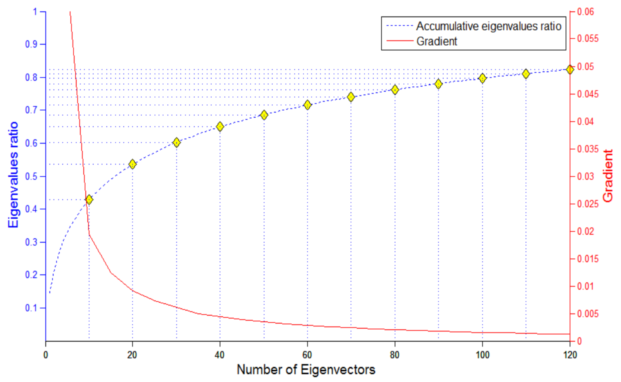


Fig. 3. Accumulative eigenvalues ratio and its gradient. The blue dashed line denotes the accumulative eigenvalues ratio while the red solid one depicts its gradient.

The upper boundary of $R_{acc}^{(k)}(k=1,2\cdots n)$ is 1.0. That means the whole energy is conserved completely. Obviously, the bigger k is set, the more computing time is required. Therefore, it is wise to set k at the place where the conserved energy is enough much and the increment rate of the energy is relatively low. As is shown in Fig. 3, the blue dashed line denotes the accumulative eigenvalue ratio. It keeps increasing, and achieves more than 70% of the whole energy when more than 70 eigenvectors are taken. However, as the read solid line shows, the increment rate of energy is quite slow after 80 eigenvectors. In other words, increasing the number of eigenvectors brought more consuming time instead of effectiveness after 80. Considering these two factors, we can finally determine 80 eigenvectors as the principal components.

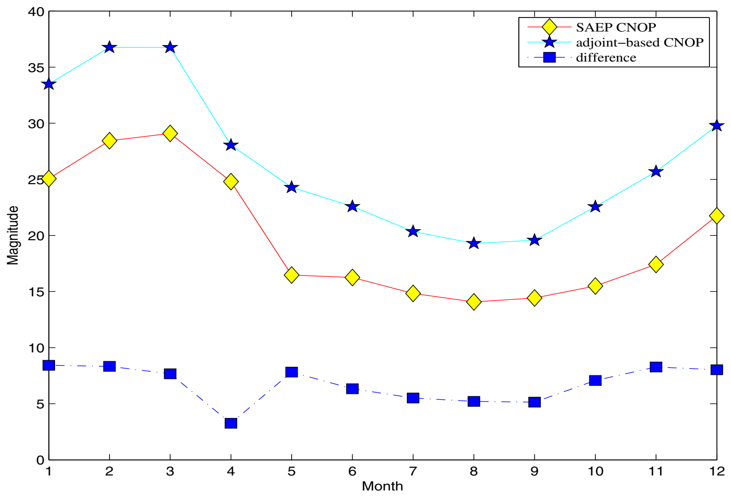


Fig. 4. The magnitudes of CNOP in SAEP vs. XD method

4.3 Comparison Analysis

The XD method is a popular solution of CNOP. Due to its high precision, it is usually treated as the benchmark. Thus, it is of necessity for the proposed method to be compared with the XD method. In this paper, we compare the magnitudes and patterns of CNOP generated by the two methods. Firstly, we set each month as the initial month and optimization time span as 9 months, respectively, and perform two methods. The magnitude of CNOP each initial month is shown in Fig 4. The red solid line with yellow diamonds represents the magnitudes of CNOP generated by SAEP while the cyan line with blue stars depicts the magnitudes of CNOP by the XD method. It is obvious that the two lines have the same trend. The CNOP goes up from January, and reaches the summit in March. Then they both keep decreasing until August. After that, they start to recover.

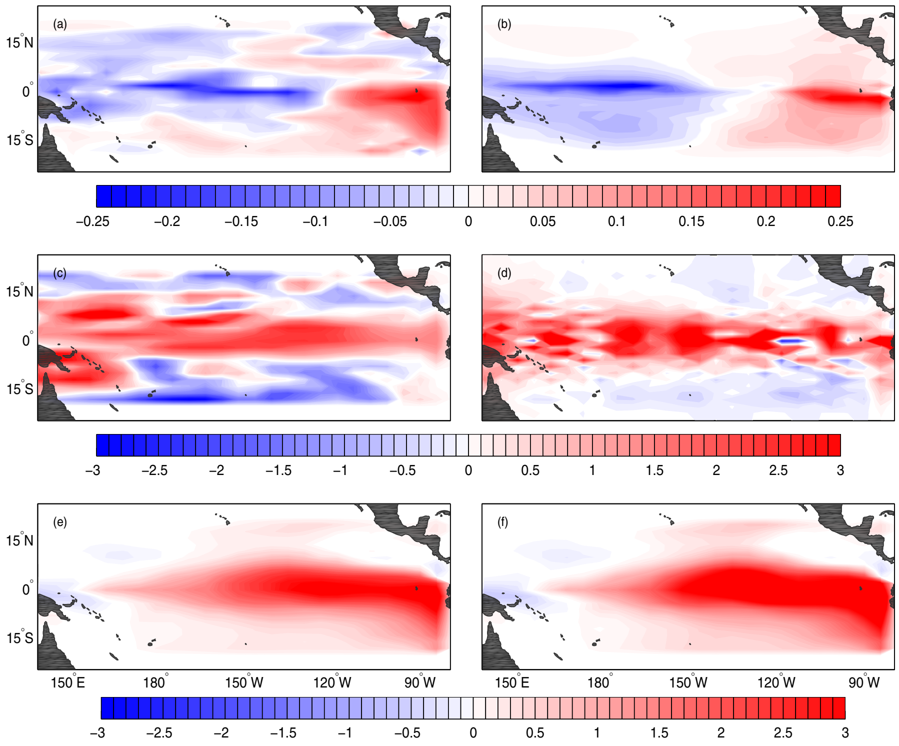


Fig. 5. CNOP pattern in January. The left column is the results from SAEP while the right column from the XD method. Three rows of subfigures are respectively the patterns of SSTA, THA and SSTA evolution in the time span of 9 months.

Besides depicting the magnitudes of CNOP in both methods, we also show the difference between them in the blue line with blue squares. The biggest difference happens in January while the smallest one in April. Without loss of generality, we choose the two months, which own the biggest and smallest difference respectively, as the examples to show the pattern comparison. As shown in Fig.5, the left column is the results from SAEP while the right one from the XD method. Three rows of subfigures are the patterns of the SSTA, THA and the SSTA evolution. Specifically, in the first row, the left SSTA keep the main large-scale characters of the right one. It is that the left part is negative while the right part is positive, which is the character of El Nino. In the third row, the patterns of SSTA evolution in both methods look the same except that the left one is a little weaker than the right one. However, the THA in the second row look different at first glance. Actually, they both have the characters that middle areas are positive while both sides of the positive area are negative. Since SAPE just take a part of all eigenvectors, its THA looks smoother than that in the XD method. Although the magnitudes of CNOP in April from both methods have biggest difference, the previous rules can be also observed in April, shown in Fig. 6. That is, the CNOP generated by SAEP look similar with that by the XD method in both cases of the biggest and smallest magnitude difference. Therefore, the conclusion can be made that SAEP can be treated as an approximate solution for the XD method.

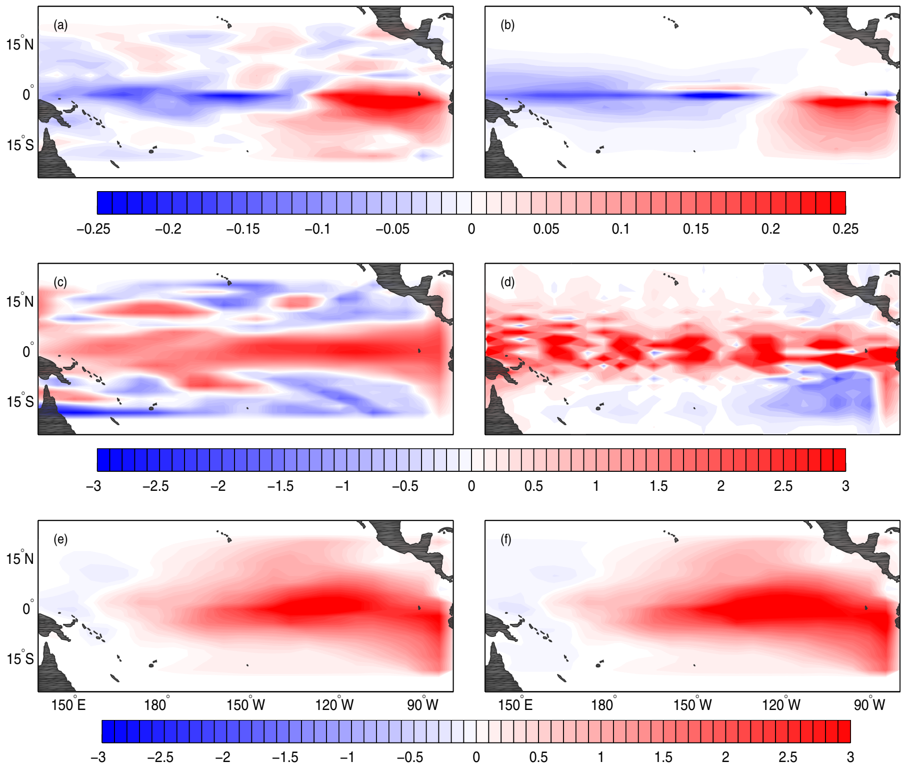


Fig. 6. CNOP pattern in Appril

4.4 Efficiency Analysis

When 80 eigenvectors are taken, the proposed method consumes about 14.9 minutes while the XD method needs 13.45 minutes with 30 initial guess fields. When just one guess field is utilized, the XD method is much faster than our method. However, the spg2 algorithm in the XD method maybe obtains the locally optimum. It cannot be guaranteed that the XD method must obtain the global optimum in the case of one guess field. The XD method thus uses 30 initial guess fields, which is of importance. Therefore, the proposed method has similar computational efficiency with the XD method.

The most advantage of our method is that it is free of the adjoint models. As discussed previously, most modern numerical models do not accompany with corresponding adjoint models. It is really a huge work to implement a new. Thus it is impossible to implement one just for CNOP. But our method can perform similar abilities with the adjoint based method, and do not need adjoint models. From this perspective, our method saves a lot of engineering time.

5 Conclusion and Future Work

This paper proposes a simulated annealing ensemble projecting method to solve CNOP. First of all, a training dataset is got from a long run of the numerical models. Then PCA is applied to the training set to obtain principal components which expand a space of the attractors. We also discuss how to choose the dimensionality k of the space. Finally, we utilize the simulated annealing algorithm to search CNOP on the space. Although our method just has similar running time with the XD method, our method is free of adjoint model. That is, we can apply the proposed method to CNOP of other numerical models without adjoint models, which improves the capability of the applications of the CNOP.

In this paper, we adopt the simulated annealing algorithm to solve CNOP. Although the speed is similar with the XD method, a parallel version of the proposed method may be more encouraging. Moreover, PCA is just one of dimensional reduction methods. Maybe the dimensionality can be further reduced by other methods. These two aspects are mainly our future work.

Acknowledgements. The authors would like to thank Hui Xu at the Institute of Atmospheric Physics, Chinese Academy of Science, for useful supports with the CNOP code of the ZC model. This work is supported by the Knowledge Innovation Program of the Chinese Academy of Sciences under Grant KZCX2-EW-201.

References

1. Lorenz, E.N.: A study of the predictability of a 28-variable atmosphere model. Tellus 17, 321–333 (1965)

2. Farrell, B.F., Moore, A.M.: An adjoint method for obtaining the most rapidly growing perturbation to oceanic flows. J. Phys. Oceanogr. 22, 338–349 (1992)
3. Samelson, R.M., Tziperman, E.: Instability of the chaotic ENSO: The growth-phase predictability barrier. J. Atmos. Sci. 58, 3613–3625 (2001)
4. Mu, M., Duan, W.S.: A new approach to studying ENSO predictability: Conditional nonlinear optimal perturbation. Chin. Sci. Bull. 48, 1045–1047 (2003)
5. Duan, W.S., Mu, M., Wang, B.: Conditional nonlinear optimal perturbation as the optimal precursors for El Nino Southern oscillation events. J. Geophys. Res. 109, 1–12 (2004)
6. Duan, W.S., et al.: Behaviors of nonlinearities modulating El Niño events induced by optimal precursory disturbance. Climate Dyn. 40, 1399–1413 (2012)
7. Mu, M., Jiang, Z.N.: A new approach to the generation of initial perturbations for ensemble prediction: Conditional nonlinear optimal perturbation. Chin. Sci. Bull. 53(13), 2062–2068 (2008)
8. Mu, M., Zhou, F.F., Wang, H.L.: A method to identify the sensitive areas in targeting for tropical cyclone prediction: conditional nonlinear optimal perturbation. Mon. Wea. Rev. 137, 1623–1639 (2009)
9. Qin, X.H., Duan, W.S., Mu, M.: Conditions under which CNOP Sensitivity Is Valid for Tropical Cyclone Adaptive Observations. Quart. J. Roy. Meteor. Soc. 139, 1544–1554 (2013)
10. Mu, M., Duan, W.S.: Conditional nonlinear optimal perturbation and its applications to the studies of weather and climate predictability. Chin. Sci. Bull. 50, 2401–2407 (2005)
11. Birgin, E.G., Martinez, J.M., Raydan, M.: Nonmonotone Spectral Projected Gradient Methods on Convex Sets. Society for Industrial and Applied Mathematics Journal on Optimization 10, 1196–1211 (2000)
12. Wang, B., Tan, X.W.: Conditional Nonlinear Optimal Perturbations: Adjoint-Free Calculation Method and Preliminary Test. American Meteorological Society 138, 1043–1049 (2010)
13. Chen, L., Duan, W.S., Xu, H.: A SVD-based ensemble projection algorithm for calculating conditional nonlinear optimal perturbation. SCIENCE CHINA (accepted, 2014)
14. Fang, C.L., Zheng, Q.: The effectiveness of genetic algorithm in capturing conditional nonlinear optimal perturbation with parameterization “on-off” switches included by a model. J. Trop. Meteor. 15(1), 13–19 (2009)
15. Zheng, Q., et al.: On the Application of a Genetic Algorithm to the Predictability Problems Involving “on-off” Switches. Adv. Atmos. Sci. 29(2), 422–434 (2012)
16. Ye, F.H., Zhang, L., Gong, X.W., Zheng, Q.: Applications of an improved particle swarm optimization to conditional nonlinear optimal perturbation. J. of Jiangnan university (Natural Science Edition) 10(4), 1–5 (2011)
17. Busetti, F.: Simulated annealing overview (2003), `http://www.geocities.com/francorbusetti/saweb.pdf`
18. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics Intell. Lab. Syst. 2, 37–52 (1987)
19. Osborne, A.R., Pastorello, A.: Simultaneous occurence of low-dimensional chaos and colored random noise in nonlinear physical systems. Phys. lett. A 181(2), 159–171 (1993)
20. Foias, C., Teman, R.: Structure of the set of stationary solution of the Novier_Stokes equations. Commun. Pur. Appl. Math 30, 149–164 (1997)
21. Metropolis, N., et al.: Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092 (1953)
22. Zebiak, S.E., Cane, M.A.: A model El Nino-Southern Osillation. Mon. Wea. Rev. 115, 2262–2278 (1987)

23. Xu, H., Duan, W.S., Wang, J.C.: The Tangent Linear Model and Adjoint of a Coupled Ocean-Atmosphere Model and Its Application to the Predictability of ENSO. In: IEEE Int'l Conf. on Geoscience and Remote Sensing Symposium, pp. 640–643 (2006)
24. Cai, M., Kalnay, E., Toth, Z.: Bred vectors of the Zebiak-Cane model and their potential application to ENSO predictions. J. Climate 16, 40–56 (2003)

Converting Ptolemy II Models to SpaceEx for Applied Verification

Shiwei Ran, Jinzhi Lin, Ying Wu, Jianzhong Zhang, and Yuwei Xu

College of Computer and Control Engineering,
Nankai University Weijin Road 94, Tianjin, 300071, China
{rsw_ok,linjinzhi}@mail.nankai.edu.cn,
{wuying,zhangjz,xuyw}@nankai.edu.cn

Abstract. Developing correct models for embedded systems requires formal verification. But it increases the burden on system designers to handle the verification techniques. In this paper, we solve this problem by providing a mapping from actor models to mathematical models suitable for verification; the conversion is automatic with minimal human intervention. We have integrated a model-designing Ptolemy II tool with verification tool SpaceEx by extending syntax with hybrid aspects. The integration of both tools and enhanced expressiveness allows Ptolemy II to model hybrid systems and get them verified by SpaceEx.

Keywords: converting algorithm, models, formal verification, Ptolemy II, SpaceEx.

1 Introduction

Design modeling and formal verification are separated in the traditional embedded system design process. Each of them has its own model representation. But in theory, they should be tightly bounded so that converting from design model to verification model is feasible.

In this paper, we present a converting algorithm which can convert a specific design model to a specific verification model, and thus after designers model systems, they can easily verify them rigorously. Little intervention is required when converting models used for verification. There is no need for designers to understand much technique of the verification model. The verification model checker can analysis the converted model and show the results.

In previous and ongoing CPS (Cyber Physical System) researches, we use Ptolemy II as the model design platform. It facilitates the design process with a component assembly framework and a graphical user interface for modeling and simulating concurrent, real-time, embedded systems [5]. On the other hand, the SpaceEx Tool [13] is a model-checker that we used for verification. It supports verification of continuous and hybrid systems which are the main target models for our CPS researches, and it facilitates the implementation of several kinds of algorithms related to reachability and safety verification and can output the analysis results in different forms clearly. Based on these, we are going to implement the so called converting algorithm that converts Ptolemy II models to SpaceEx models for applied verification.

This paper is organized as follows. Related works are introduced in section 2. In section 3, the model structures for Ptolemy II and SpaceEx are defined respectively. Next, in section 4 we present the execution semantics of models and the core of the converting algorithm. Then we conduct two case studies in section 5. Finally, we conclude this paper in section 6.

2 Related Works

In Ptolemy II, researchers previously focused on verification in DE and SR domains to perform the conversion process [1]. For the SR domain, they translate models into formats acceptable by Cadence SMV (also NuSMV) [12], and DE domain can be viewed as a generalization of the SR domain, with a global clock visible to the whole system, they translate models into formats acceptable by RED [18] for it. Comparing with DE, the CT domain concerns more systems that can be modeled using ordinary differential equations (ODEs) in continuous-time semantic. Reference [2] integrated a model-checking UPPAAL tool (Timed Automata) with HyTech model-checker (Hybrid Automata) by extending UPPAAL syntax with hybrid aspects.

Several of verification tools are emerged to check real-time, embedded systems. HyTech [15] is an automatic tool for the analysis of embedded systems. It computes the condition under which a linear hybrid system satisfies a temporal requirement. Passel [16] is a tool for performing verification of networks composed of interacting hybrid automata. The PHAVer (Polyhedral Hybrid Automaton Verifier) [14] is a tool for verifying safety properties of hybrid systems. The UPPAAL [17] verifier uses model-checking techniques, that is, given a model with initial states and a formula, it traverses the whole state-space to find whether the model match the formula.

3 Model Abstract Structures

3.1 Ptolemy II Models and Designs

Ptolemy II offers an infrastructure for implementations of a number of components called actors. Now we present the following abstract syntax of an actor to further our research.

Definition 1. An actor is a tuple A = <E, P, Para, R> [1], where

- *E* is a finite set of (inner) actors.
- *P* is the set of ports.
- *Para* is the set of parameters.
- *R* are the channels identifying a source port and a target port between (1)two inner actors' ports or (2) a port of the actor and a port of the actor's inner actor.

A system is merely a composite actor; its hierarchical structure is defined by the recursive definition of its inner actor set. In Fig.1, the system is represented as $A_0 = \{E_0, P_0, Para_0, R_0\}$, where $E_0 = \{A1, A2, A3, D1\}$, $P_0 = \Phi$, $R_0 = \{(P_0, P_1), (P_2, P_3)\}$. A_1 is an atomic actor, while A_2 is composite (in fact it's an instance of a modalmodel [7]). The system consists of an inner modalmodel actor, whose tuple A has some differences from the system A_0.

Hybrid systems are models that combine continuous dynamics with discrete mode changes [6]. They are created in Ptolemy II by creating ModalModels, a ModalModel actor contains a *ModalController* and a set of *Refinement* actors that model the refinements associated with states. Modalmodels can be constructed with other domains besides CT [7], but here we only concentrate on CT. In Fig.1, A_2 is a modalmodel, it can be represent as $A_2 = \{E_2, P_2, Para_2, R_2\}$ where $E_2 = \{A_4, A_5\}$, A_4, A_5 are refinement states in modalmodel A_2. Each refinement is required to have its own director and inner actors. $P_2 = \{p_1, p_2\}$, R_2 are still channels but their sources and targets are changed to those states with refinements, these channels describe discrete jumps between continuous dynamic states, rather than the channels between actors' ports.

3.2 Models Used in SpaceEx for Verification

Hybrid automata is a widely used formalism for describing systems with both continuous and discrete dynamics [9, 11]. Verification tool SpaceEx has implemented hybrid automata and some algorithms that compact formal description of the complex behavior of such systems.

For models that made up of one or several modalmodels, each modalmodel corresponds to a component, named *base component*, which consists of some attributes like locations and transitions. A *network component* consists of one or more instantiations of other components (base or network) and it corresponds to a set of hybrid automata in parallel composition [4]. Here we introduce the SpEx structure used by the model checker.

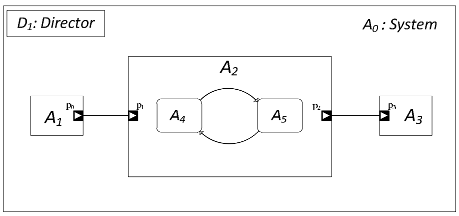


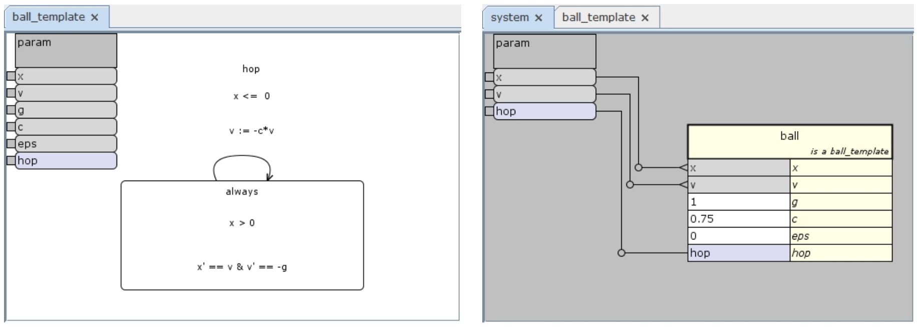
Fig. 1. A system featuring actors and directors in Ptolemy II

Definition 2. A SpEx structure is a tuple $S = \{CPB_1, CPB_2,\ldots; CPNT_1, CPNT_2,\ldots\}$, where $CPB_i = \{ PARA_B, LOC, TRAN\}$ is a base component in the SpEx structure with the following constraints:

- $PARA_B$ = {$para_1$, $para_2$,…} is the set of formal parameters; each parameter is of the form $para_i$ ={name, type, dynamics}. A formal parameter may be: (1) a continuous variable (2) a constant (3) a synchronization label.
- LOC = {loc_1, loc_2,…} is the set of locations within the base component; each location is of the form loc_i ={name, invariant, flow}, location (state) has a unique name, a invariant constraint on $PARA_B$ and a flow containing some differential equations for describing continuous dynamics evolution within the location.
- TRAN = {$tran_1$, $tran_2$,…} is the set of jumps between locations; each transition is of the form $tran_i$ = {source, target, label, guard, assignment}. A transition has a source location and a destination location, and has a label as its name, a constraint condition and an assignment to modify the values of continuous variables when the transition triggered.

Just as the base component, the network component can be defined as $CPNT_i$ = {$PARA_N$, BIND}, it is an instantiation of a base component with the following constraints:

- PARAN = {para1, para2,…} is the set of formal parameters; each parameter is of the form parai ={name, type, dynamics}. It only contains the continuous variable and synchronization label of its inner components.
- BIND = {bind1, bind2,…} is the set of initial variables or labels; each bind has the form that bindi ={component, MAP}, component indicates which base component is to bind, and MAP = {map1, map2 …} is the set of mapi = <key, value>, which give values for the parameters in the binding base component. The value of a mapi can be an explicit value or a parameter defined in PARAN of the network component.



(a) (b)

Fig. 2. The Bouncing Ball's base component (a), and the network component (b)

Fig.2 shows the modular model of a Bouncing Ball in SpaceEx graphical editor. Its SpEx structure can be depicted as: S = {ball_template; system}. It only contains a base component—*ball_template* and a network component—*system* which is the instantiation of *ball_template*. In the base component, *ball_template* contains six parameters, one location and one transition, $PARA_B$ = {{x, real, any}, {v, real, any}, {g, real, const}, {c, real, const}, {eps, real, const}, {hop, label, -}}; LOC={{always, x>=0, x'==v&v'==-g}}; TRAN = {{always, always, hop, x<=eps & v<0,v:=-c*v}}. In the network component (b), it only contains one instantiation of the base component *ball_template*. Its PARA_N = {{x, real, any}, {v, real, any}, {hop, label, -}}, BIND = {{ball_template, MAP}}, and MAP = {{x, x}, {v, v}, {g, 10}, {c, 0.75}, {esp, 0.01}, {hop, hop}}. For the variables and label of base component *ball_template*, MAP maps them to their corresponding parameters in the network component. But for constants, MAP assigns them constant values as the initial values.

4 CT Model Analysis and Conversion

In Ptolemy II, directors control the execution order and domain polymorphic of actors. To convert the CT domain models into SpEx structure, we should understand the model executing process under CT semantics and abstract interpretation of the modalmodel converting.

4.1 Execution Semantics of Modalmodel under the CT Domain

In CT domain, a modalmodel is constructed in a modal model actor having the continuous director as local director. This director mediates the interaction with the outside domain, and coordinates the execution of the refinements with the mode controller [7].

The execution semantics of modalmodel are controlled by its local director; it has the key flow of control methods:

- *prefire():* test precondition to fire(); a modalmodel can always prefire().
- *fire():*

1. The continuous director transfers the input tokens to the mode controller and to the refinement of its current state.
2. The mode controller examines the current state, if its preemptive transition can be triggered, execute the choice actions of the transition, and skip the (3) and (4).
3. Fire the refinement of the current state.
4. The mode controller examines the non-preemptive transition, if it can be triggered, execute the transition.
5. Any output token produced by mode controller or refinement is transferred to the outside domain.

- *postfire():* if a transition is triggered in fire(), it executes the choice actions of the transition to update the state and its attributes.

4.2 Abstract Interpretation of the Converting Algorithm

Converting a common composite actor in Ptolemy II to the SpEx structure needs analysis of the model's actors, especially modalmodel actors. In theory, a modalmodel actor can be represented as A_{mm} = (E, P, Para, R) = (S, P, Para, Tran).

- *S* is the set of states in the modalmodel. The modalmodel is a finite-state machine controller actor, but inside each state of the FSM is a refinement model. The refinement also can describe as (E, P, Para, R), where E is the set of the actors in this model, like *Integrator*, *Const*, *LevelCrossingDetector* and so on. Each refinement model is required to have its own director [8].
- *P* is the set of ports in the modalmodel. For the same ports of the actor, all states share the same instantaneous value of the port.
- *Para* is the set of parameters in the modalmodel. It contains elements of the attributes in the modalmodel controller configuration, initial values of source actors (like *Const*) and the variables in transition properties.
- *Tran* is the set of transitions between states in the modalmodel. Elements in *Tran* contain *guard*s which specify the transition conditions and *setActions* which assign new values for parameters when necessary.

By prescaning the models, we can convert a modalmodel to a base component and create an instantiations of the component as a network component. The algorithm follows:

```
Convert_ModalModel_To_CPB_CT( actor $A_{mm}$ ){
  /* $A_{mm}$ = (S, P, Para, Tran) */
  Let $PARA_B = \Phi$ be the set of parameters in CPB
     (1) $PARA_B = PARA_B \cup \{v\}, \forall v \in P$,
          $v_{name} = P_{name}$, $v_{type}$ = real, $v_{dynamic}$ = any;
     (2) $PARA_B = PARA_B \cup \{c\}, \forall c \in Para$,
          $c_{name} = Para_{name}$, $c_{type}$ = real, $c_{dynamic}$ = const;
     (3) $PARA_B = PARA_B \cup \{l\}, \forall l \in Tran$,
          $l_{name} = Tran_{name}$, $l_{type}$ = label, $l_{dynamic}$ = - ;
  Let $LOC = \Phi$ be the set of locations in CPB
     $\forall s \in S$, LOC = LOC $\cup$ {loc},
          $loc_{name} = s_{name}$;
          $loc_{invariant} = \overline{Tran_{guard}}$, where
               $Tran_{guard}$ is the transition guard expression whose source is the *loc*;
               $\overline{expr}$ means the inverse proposition of *expr*.
          $loc_{flow} = s_{derivative.1}$&,..., $s_{derivetive.n}$, where
               $s_{derivative.i}$ represents the *i-th* derivative relation in the location *s*,
               which is further described in the next paragraph.
  Let $TRAN = \Phi$ be the set of transitions in CPB
      $\forall tran \in Tran$, TRAN = TRAN $\cup$ {t},
          $\exists$ loc1 $\in$ LOC s.t. $loc1_{name} = tran_{source}$, then $t_{source} = loc1_{name}$;
          $\exists$ loc2 $\in$ LOC s.t. $loc2_{name} = tran_{target}$, then $t_{target} = loc2_{name}$;
          $t_{label}$ = [ $l \in tran_{guard}$.xx_isPresent] || [$l \in tran_{destination.port}$];
          $t_{guard} = tran_{guard.1}$&,..., $tran_{guard.n}$, where
               $tran_{guard.i}$ represents the *i-th guard* expression of the transition.
          $t_{assignment} = tran_{set.1}$&,.., $tran_{set.n}$, where
```

```
            $tran_{set.i}$ represents the *i-th setActions* of the transition.
    Return ($PARA_B$, LOC, TRAN).
}
CPBs_INSTANTIATED_TO_CPNT_CT ( ) {
    Let $PARA_N = \Phi$ be the set of parameters in CPNT.
    Let BIND = $\Phi$ be the set of initial variables and labels.
    $\forall CPB \in SpEx$
        /* CPB = ($PARA_B$, LOC, TRAN) */
        $\forall$ pb $\in PARA_B$ s.t. ($pb_{dynamics}$=any || $pb_{type}$=label), $PARA_N = PARA_N \cup \{p\}$,
            $p_{name} = pb_{name}$;
            $p_{type} = pb_{type}$;
        BIND = BIND $\cup$ {bind},
            $bind_{component} = CPB_{name}$;
            Let MAP = $\Phi$, MAP = MAP $\cup$ {map}
                (1) $\forall$ v_l $\in PARA_B$ &$v\_l_{dynamic}$! = const,  $\exists$ p $\in PARA_N$ s.t. $p_{name}$=$v\_l_{name}$,
                    $map_{key} = v\_l_{name}$, $map_{value} = p_{name}$;
                (2) $\forall$ c $\in PARA_B$ & $c_{dynamic}$ = const,  $\exists$ p $\in PARA_N$ s.t. $p_{name}$=$v\_l_{name}$,
                    $map_{key} = c_{name}$, $map_{value} = p_{initialValue}$;
            $bind_{MAP}$ = MAP.
    Return ($PARA_N$, BIND).
}
```

Above methods lead to a converting algorithm for a CompositeActor with one or more modalmodels to SpEx structure. A feasible SpaceEx structure contains base components part and instantiation part by the network components. In the first algorithm, the derivative relation is acquired from the specific actor *Integrator* in Ptolemy II (Fig.3), the actor has an input port *derivative* and an output port *state*, the value in the *derivative* side is the derivative of the *state* one. We traverse the port list of the actor, then acquire the values from the appointed ports and make up the derivative formula that has the following format: *state' == derivative*.

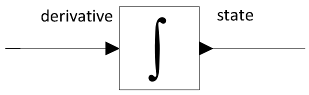


Fig. 3. Integrator Actor in Ptolemy II

4.3 Interactions between Modalmodels

A modalmodel corresponds to a base component. When converting a Ptolemy II model with two or more modalmodels to a SpEx structure, the interactions between modalmodels must be considered. A modalmodel may have ports to communicate with other modalmodels. If two ports respectively in two different modalmodels connect with each other, these ports have the same significance for the system and the signal value derived from the ports keep synchronized in the system.

Just as Fig.4 shows, in MM_2, the guard expression of transition from $state_1$ to $state_2$ is "P_2_*isPresent*", once P_2 signal is present, the transition will be triggered. The P_2 signal derives from port P_1 in MM_1, so in MM_1, there must have a transition that triggers the output of P_1. The transitions in MM_1 and MM_2 have a synchronized relation.

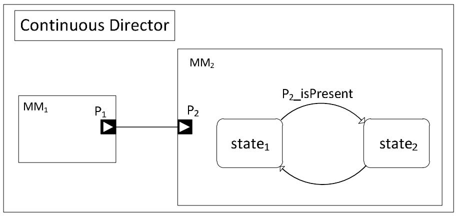


Fig. 4. Interactions between modalmodels

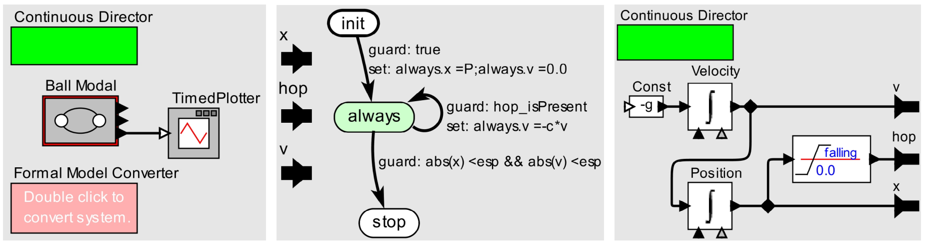
Transitions corresponding to labels in SpaceEx, so it offers a mechanism called synchronization labels to handle the composition of two automata with HIOA-style parallel composition [3]. That's to say, label parameters from different base components to be declared as non-local and to mapped together when instantiated in a network component can be synchronized. So for the interactions between modalmodels, complying with this synchronization mechanism, the converting algorithm respectively defines non-local label parameters and assigns them to the linked synchronized transitions' labels in the corresponding base components, then carefully maps these label parameters together when instantiating to network component.

5 Case Studies

Our work is intended to provide an auto-converting method to transform a model which has CT semantic designed by Ptolemy II to the SX format [3] (the modeling language of verification tool SpaceEx). In this section, we present two typical scenarios to validate our method as well as to show the details of the transforming process.

5.1 The Bouncing Ball

Fig.5 shows the model simulating the process of a falling ball in the CT domain [7].

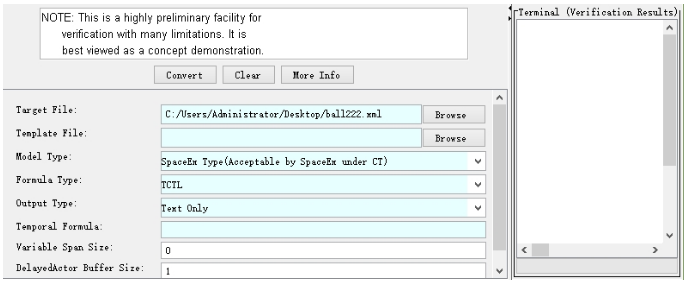


(a) (b) (c)

Fig. 5. (a) is the Bouncing Ball model in Ptolemy II, (b) is the Ball modalmodel, (c) is inner refinement of state "*always*"

The ball begins falling at the height of an initial position. When it downs to 0 and starts to bounce up, its velocity could loss in a certain proportion. With continually bouncing and falling, the ball turns to stop when its position and velocity are lower than a threshold.

This model has three states: *init*, *always* and *stop*. States *init* and *stop* are special supporting startup and stopping of simulation in Ptolemy II, thus they are not under consideration in the converting process. The transition with label guard *hop_isPresent* can take the automaton from the *always* state to the next once the *hop* signal is presented. This model is a typical one base component in SpEx structure, it's Sball = {CPB1; CPNT1} = {ball_template; system}. For our convenience to make convert, Ptolemy II offer a convert interface as Fig.6. But to realize our design, we must provide our Model Type: SpaceEx Type (Acceptable by SpaceEx under CT). Only choose this type, can we convert the model to SpaceEx objective file.


NOTE: This is a highly preliminary facility for verification with many limitations. It is best viewed as a concept demonstration.

Target File:	C:/Users/Administrator/Desktop/ball222.xml
Template File:	
Model Type:	SpaceEx Type(Acceptable by SpaceEx under CT)
Formula Type:	TCTL
Output Type:	Text Only
Temporal Formula:	
Variable Span Size:	0
DelayedActor Buffer Size:	1

Fig. 6. The converter interface of SpaceEx in Ptolemy II

Base Component and Converting.

Every base component has a name attribute tagged as "*id*" which defined by SX format like:

```
<component id="ball_template">
  …
</component>
```

We get the id of the component from the modalmodel's name directly. In this model, the position *x* and velocity *v* is the main variables to describe the movement of the ball and we get them from the ports contained in the *always* refinement. The acceleration *g*, the scale parameter *c* and the threshold value *esp* are treated as constant as they are configuration parameters in Ptolemy II. The label parameter *hop* is a signal which acquired from the transition guard condition.

```
<param name="x" type="real" dynamics="any" />
...
<param name="g" type="real" dynamics="const" />
...
<param name="hop" type="label" />
```

Corresponding to the *always* state in Ptolemy II, a location named *always* is gained in SpEx structure. The guard expression of the transition whose source is linked to always state is "*hop_isPresent*", as guard expression of the form "*XX_isPresent*" is related to the port *XX*, here we make a contribution to analyse the inner execution of port *hop*, and then find that "*hop_isPresent*" is equal to "x<=0". Thus, we have "x>0" as the *invariant* of *always* location. We get the *flow* of the location by analyzing the *Integrator* actors in the refinement: two integrator actors in the model correspond to two derivative formulas, so as *v* is the derivative of *x* and –*g* the derivative of *v*.

```
<location id="1" name="always">
  <invariant>x &gt; 0</invariant>
  <flow>x' == v &amp; v' == -g</flow>
</location>
```

Excluding the special two states *init* and *stop*, there is only one transition in this model, accordingly, converting algorithm generates one transition. According to above, the transition's guard is "x<=0"; and its assignment is the same as setActions in Ptolemy II. In addition, the *source* and *target* of this transition can be easily determined as both ids of *always* location, which is "1".

```
<transition source="1" target="1">
  <label>hop</label>
  <guard>x &lt;= 0</guard>
  <assignment>v := -c*v</assignment>
</transition>
```

Network Component and Converting.
As indicated in section 3.2's algorithm CPBs_INSTANTIATED_TO_CPNT_CT, for a base component, only parameters with *any* dynamics or with *label* type should be declared in PARA_N, thus we have the following parameters in the generated network component:

```
<param name="x" type="real" dynamics="any" />
<param name="v" type="real" dynamics="any" />
<param name="hop" type="label" />
```

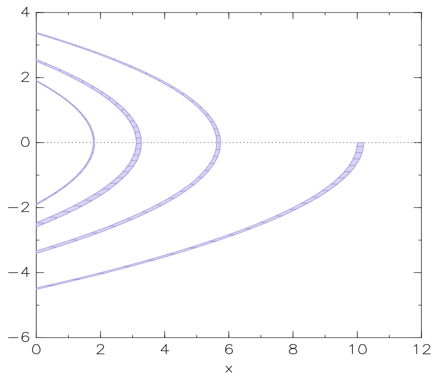


Fig. 7. The reachability analysis result of the Bouncing Ball System by SpaceEx

In this network component, when binding the base component ball_template's parameters, variables *v*, *x* and *label* hop are bound to their corresponding parameters just mentioned above, while *g*, *c* and *esp* are respectively bound to numeric values 10, 0.75 and 0.01.

```
<bind component="ball_template" as="ball ">
      <map key="x">x</map>
   <map key="v">v</map>
   <map key="g">10</map>
   <map key="c">0.75</map>
   <map key="eps">0.01</map>
   <map key="hop">hop</map>
</bind>
```

Fig.2 actually is the converting result of the Bouncing Ball system, which can be directly opened in SpaceEx Model Editor. After this, the system can be formally validated by SpaceEx; the input is the file our converting algorithm has produced. The validation result of this case is shown in Fig.7, which gives all the reachable states the bouncing ball can go. We can see that our generated file can be properly analysed by SpaceEx tool for validation.

As this case consists of only a single automaton, it may not indicate the interactions between different automatons. So the situation changes when we look at systems made up of more than one automaton, which is the topic of the next section.

5.2 The Water Tank System

In this case, we demonstrate a complex system shown in Fig.8. It consists of two interacted modalmodels. The modalmodel *WaterTank* simulates two leaking water tanks and one injection plant, where x_1, x_2 are the current water levels, r_1, r_2 are the leaking rates and *w* is the injection rate. Only one tank can be injected at a time, which is controlled by the *Controller* modalmodel. At the beginning, one of the two tanks is both leaking and injecting while the other tank is only leaking. When the only leaking tank's water level becomes less or equal to a threshold value c, the *Controller* produces a command to switch the injection plant to fill the leaking tank. In this way, the injection plant is controlled to switch back and forth between the two leaking tanks. As we can image, if *w* is less enough, the two tanks could be finally dried up. It is supposed that SpaceEx could validate this situation.

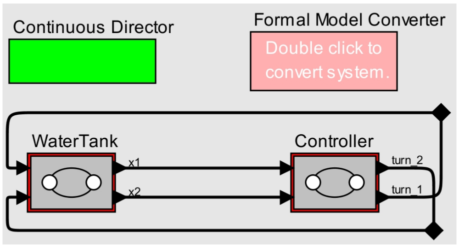
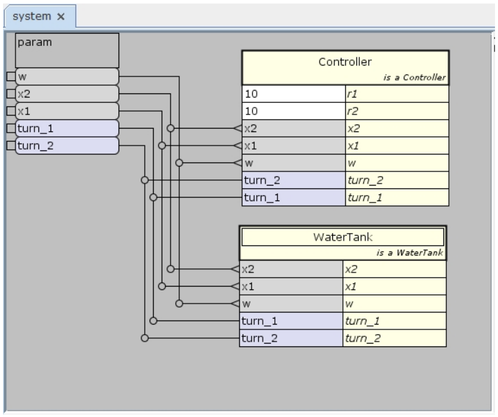


Fig. 8. The Water Tank System model in Ptolemy II

Interactions between WaterTank and Controller.
In the generated SpEx structure, two CPBs named *WaterTank* and *Controller* and a CPNT named *system* have been produced. In Ptolemy II, *WaterTank* and *Controller* is connected by ports *turn_2*, *trun_1*, x_1 and x_2. X_1 and x_2 are dynamic variables which are declared as parameter with any dynamic and real type. For the connections *turn_1* and *turn_2*, however, "*turn_1_isPresent*" and "*turn_2_isPresent*" appear in *WaterTank* modalmodel's state transitions, they should be treated as synchronization labels. Fig.9 shows generated result of this case. In the two CPBs *WaterTank* and *Controller*, label parameters *turn_1* and *turn_2* are respectively declared as non-local, and they are both mapped to the corresponding label parameters defined in the *system* CPNT. According to section 4.3 and its generated result, our converting algorithm realizes the conversion of the interaction between modalmodels in Ptolemy II to the interaction between base components in SpaceEx.



w
x2
x1
turn_1
turn_2

Controller
is a Controller

10	*r1*
10	*r2*
x2	*x2*
x1	*x1*
w	*w*
turn_2	*turn_2*
turn_1	*turn_1*

WaterTank
is a WaterTank

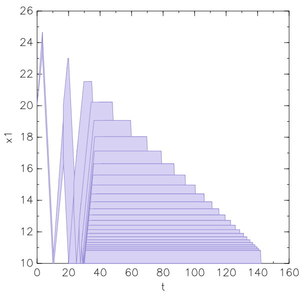
x2	*x2*
x1	*x1*
w	*w*
turn_1	*turn_1*
turn_2	*turn_2*

Fig. 9. The network component view of the Water Tank System

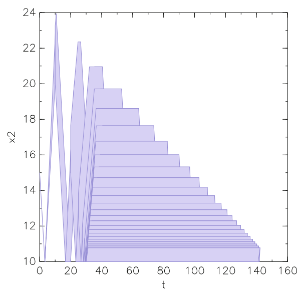
Reachability Analysis.
After converting the Ptolemy II models to SpEx structure in SX format, we can check the security of the model using SpaceEx validation tool by computing the sets of reachable states of the system [10].

In this case, SpaceEx can help us to get the rational value of the inject rate w according to the reachability analysis. Here, we initialize the parameter values as: x_1=20, x_2=15, c_1=c_2=10, r_1=2, r_2=1.5. Fig.10 shows the results of different ranges of rate w by reachability analysis. (a), (b) and (c) are the results when w<3.5, the water levels x_1, x_2 of tank1 and tank2 are varying over time and both could eventually converge to 10. If $x_1 \leq (c_1+\varepsilon)$ and simultaneously $x_2 \leq (c_2+\varepsilon)$ are regarded as dangerous states, then we could draw the conclusion from (c) that w<3.5 could lead to the dangerous states which are undesired. When w=3.5(d), the same as the sum of r_1 and r_2, the system's whole water injecting and ejecting rates are kept balanced, dangerous states are unreachable. w>3.5(e) guarantees the sum of the two water levels increase, and of course dangerous states are not reachable.

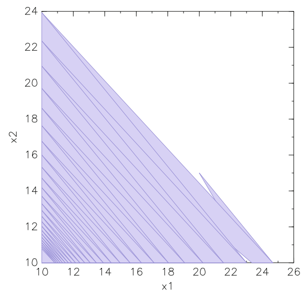
As these profound reachability analysis results reveal, our converting algorithm generates the correct SpEx structure in SX format with complex interactions among modalmodels, through accurately dealing with label synchronization provided by SpaceEx.



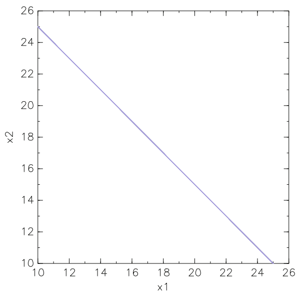
(a) 3≤w≤3.4, x_1 vs. *time*.



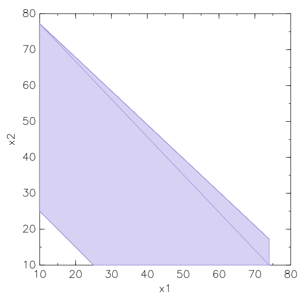
(b) 3≤w≤3.4, x_2 vs. *time*.



(c) 3≤w≤3.4, x_2 vs. x_1.



(d) w = 3.5, x_2 vs. x_1.



(e) 3.5<w≤3.6, x_2 vs. x_1.

Fig. 10. The reachability analysis results of the Water Tank System by SpaceEx

6 Conclusion

For the purpose to simplify embedded systems design process and reduce designing burdens, we introduce an automatic converting approach from the designed models in Ptolemy II to verified models in SpaceEx in this research. By working on the file formats and model structures of Ptolemy II and SpaceEx, the abstract semantics of models in these two tools are defined. After carefully contrasting the abstract semantics and working out their mapping relations, the converting algorithm is proposed. In the following work, we have verified this algorithm through two case studies which respectively with a single modalmodel and two modalmodels with interactions. By applying this approach to what, hybrid models for embedded systems can be designed and simulated in Ptolemy II efficiently, then with minimum human intervene, auto-converted to SpaceEx model for formal verification, which was previously not possible.

Acknowledgments. This work was supported by the Research Fund for the Doctoral Program of Higher Education of China (No. 20110031110026 and No. 20120031110035), and the Fundamental Research Funds for the Central Universities - the Science Foundation for Young Scholars of Nankai University (No. z1a2035946).

References

1. Cheng, C.P., Fristoe, T., Lee, E.A.: Applied verification: The ptolemy approach. Technical Report UCB/EECS-2008-41, EECS Department, University of California, Berkeley, April 2008 (2008)
2. Mufti, W.A., Tcherukine, D.C.: Integration of model-checking tools: from discrete to hybrid models. In: IEEE International Multitopic Conference (INMIC 2007), pp. 1–4. IEEE (2007)
3. Cotton, S., Frehse, G., Lebeltel, O.: The SpaceEx modeling language (2010)
4. Frehse, G.: An introduction to spaceex v0. 8 (2010)
5. Lee, E.A., Davis, I., Muliadi, L., Neuendorffer, S., Tsay, J.: Ptolemy II, Heterogeneous Concurrent Modeling and Design in Java. DTIC Document (2001)
6. Alur, R., Courcoubetis, C., Henzinger, T., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. In: 11th International Conference on Analysis and Optimization of Systems Discrete Event Systems, pp. 329–351. Springer, Heidelberg (1994)
7. Brooks, C., Lee, E.A., Liu, X., Neuendorffer, S., Zhao, Y., Zheng, H.: Heterogeneous concurrent modeling and design in java (volume 3: Ptolemy ii domains). EECS Department, University of California, Berkeley, UCB/EECS-2008-37 (2008)
8. Brooks, C., Lee, E.A., Liu, X., Neuendorffer, S., Zhao, Y., Zheng, H., Bhattacharyya, S.S., Cheong, E., Davis, I., Goel, M.: Heterogeneous concurrent modeling and design in java (volume 2: Ptolemy ii software architecture). DTIC Document (2008)
9. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical computer science 138, 3–34 (1995)

10. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011)
11. Alur, R.: Formal verification of hybrid systems. In: 2011 Proceedings of the International Conference on Embedded Software (EMSOFT), pp. 273–278. IEEE (2011)
12. Bae, K., Ölveczky, P.C., Feng, T.H., Tripakis, S.: Verifying ptolemy II discrete-event models using real-time maude. In: Breitman, K., Cavalcanti, A. (eds.) ICFEM 2009. LNCS, vol. 5885, pp. 717–736. Springer, Heidelberg (2009)
13. SpaceEx | State Space Explorer, `http://spaceex.imag.fr/`
14. Frehse, G.: PHAVer: Algorithmic verification of hybrid systems past hyTech. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 258–273. Springer, Heidelberg (2005)
15. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: HyTech: A model checker for hybrid systems. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 460–463. Springer, Heidelberg (1997)
16. Passel, `https://wiki.cites.illinois.edu/wiki/display/MitraResearch/Passel`
17. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer (STTT) 1, 134–152 (1997)
18. Wang, F.: Efficient data structure for fully symbolic verification of real-time software systems. In: Graf, S. (ed.) TACAS 2000. LNCS, vol. 1785, pp. 157–171. Springer, Heidelberg (2000)

Research on Interest Searching Mechanism in SNS Learning Community

Renfeng Wang, Junpei Liu, Haining Sun, and Zhihuai Li

School of Information Science and Technology,
Dalian Maritime University,
Dalian, Liaoning, China
renfeng@dlmu.edu.cn

Abstract. The SNS learning community searches resources mainly based on the learners' interests. It significantly influences the learners' positivity of self-study whether the interest searching efficiency is high or not. However, the existing interest-based searching mechanisms are not comprehensive in node interest expressions and seem to be unduly complex in the calculation of the relevant degrees between the learners' interests, which lead to low searching efficiency. Aimed at improving these deficiencies, it proposes more accurate methods of node interest expressions. Considering both efficiency and comprehensiveness of the calculation of relevant degree between node interests, it forms nodes with similar interests into effective interest domains to realize high interest searching efficiency. The comparisons of the Matlab simulation experiment results demonstrate that the improved searching mechanism can greatly promote the searching performance.

1 Introduction

Conventional teaching mode is teacher- centered[1], which is unidirectional teaching with few interactions, it is difficult to stimulate the enthusiasm of the learners' self-learning.While in SNS learning community [2,3,4,5,6], all members are equal, anyone can both acquire knowledge as student and teaching as teacher at the same time. In particular, searching resources is based on learners' interests, which motivates the learners better. This paper is Aiming at improving interest-based searching efficiency in SNS learning community. The existing searching mechanisms are not fully considered the statistical properties of node interests, as well as single-angle expression of node interests can not fully express learners' interests, which leads to both complicated and inefficiency while calculating correlation degree between interest nodes. This paper propose a more accurate way for node interest expression, which conduct interest search in interest cluster or interest domain that is consist of nodes with similar interests on the premise of making comprehensive consider for computational efficiency and comprehensiveness while computing similarity of interest nodes. By which provides an efficient interest-based searching mechanism for SNS learning community.

2 Network Topology of SNS Learning Community

The basic unit of network topology of SNS learning community is interest domain, which is formed by nodes with similar interests and hybrid structure is adopted[7]. Distribute the high performance super nodes to the center of interest domain, and the rest as leaf nodes to the periphery. In the interest domain, an interest cluster consists of a super node and multiple leaf nodes form a interest cluster, so that interests are much better organized. Topology of interest domain is showed in Figure 1:

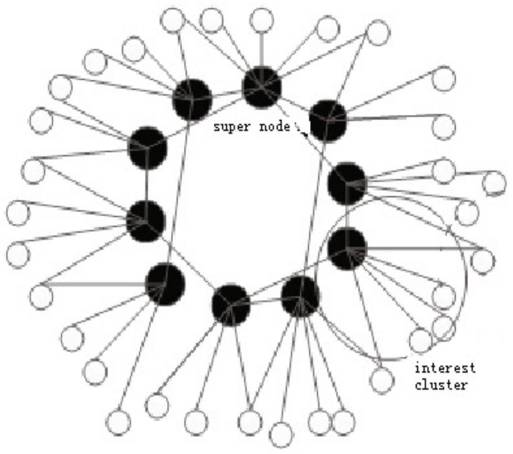


Fig. 1. Network topology of interest area

3 Joins Interest Domain

3.1 Find Similar Interests

Learners' interests are showed by the topics of their document sets.

Definition 1. Learners' interests in vector ($\boldsymbol{V_{si}}$:student interest vector)

$$\boldsymbol{V_{si}} = (v_1, v_2, \ldots v_n) \tag{1}$$

Learners' interests are divided into n types, and the type i is called Interest[i](i=1 to n),$\sum_{i=1}^{n} v_j = 1(j = 1, 2 \ldots n)$.

Definition 2. Entry Selectesentry select: Choose some relatively important entries, rather than all the entries in the entire document set.

$$es(t_i, C) \geq DFT \tag{2}$$

$es(t_i, C)$ is the document frequency of the entry t_i in the document C, and DFT is the preset threshold value.

Definition 3. Priori probability estimates of documentation set entry: It mainly refers to the probability model of the document themes, namely, a probability distribution $(\hat{p}(t_1 \mid M_T), \hat{p}(t_2 \mid M_T), \cdots \hat{p}(t_m \mid M_T))$ on entry set $\{t_1, t_2, \cdots t_m\}$.

$$\hat{p_{ev}}(t_i|M_T) = \frac{\sum_{d_j \in C} p(t_i|M_{d_j})}{|C|_d} \tag{3}$$

C is the document set of theme T,$|C|_d$ is the number of documents in C,d_j is a document in C.

Definition 4. Themes SetTthemes: Node has all the themes included in the set of document C.

$$T = \{t_1, t_2, \cdots t_m\} \tag{4}$$

Definition 5. Probability of document set theme model $(p(t_i|M_T))$:

$$p_{ev}(t_i|M_T) = \frac{\sum_{d_j \in C} p(t_i|M_{d_j})}{|C|_d} \tag{5}$$

get $(p(t_1 \mid M_T), p(t_2 \mid M_T), \cdots p(t_m \mid M_T))$, descending sort.

Definition 6. k-high frequency word vector $(pk(t_i|M_T))$: calculate the maximum frequency of the former k nouns appear in vector component of $(p(t_1 \mid M_T), p(t_2 \mid M_T), \cdots p(t_m \mid M_T))$, and use the sum of this k noun frequencies as the base $\sum_{j=1}^{n}(pk(t_i|M_T))$. Calculating the probability of higher frequency nouns which belong to the same category of interest, the form of statistics is $\sum_{w_m \in interest[j]} (pk(t_i|M_T))$, w_m represents the m-th noun of the highest frequency of the top k, $pk(t_i|M_T)$ represents frequency of w_m occurrences in the document. Finally, get the learners' expression of interest $\boldsymbol{V_{si}} = (v_1, v_2, \cdots v_n)$:

$$v_i = \frac{\sum_{w_m \in interest[j]} (pk(t_i|M_T))}{\sum_{j=1}^{n}(pk(t_i|M_T))} \tag{6}$$

Definition 7. Similar interest (Sim: Similar interest):

$$Sim(\boldsymbol{V_{si1}}, \boldsymbol{V_{si2}}) = \frac{\sum_{k=1}^{n}|v_{si1k} - \bar{v_{si1}}| \cdot |v_{si2k} - \bar{v_{si2}}|}{\sqrt{\sum_{k=1}^{n}(v_{si1k} - \bar{v_{si1}})^2} \cdot \sqrt{\sum_{k=1}^{n}(v_{si2k} - \bar{v_{si2}})^2}} \tag{7}$$

$Sim(\boldsymbol{V_{si1}}, \boldsymbol{V_{si2}})$ means the greater the correlation coefficient, the interest is more similar between the two learners.

3.2 Joins Interest Domain

For the process of node joins the interest domain:

(1) Prior to joining SNS learning community node p^* initialized, the main task is marshaling local set of documents, generating node expression, achieving the purpose of initializing the library of node information;
(2) If node p^* is the first node adds to the network, then specify it as the super node, and create a new interest domain;
(3) If node p^* already knew an exists node before joining the SNS learning community, then it can pre-establish connection with the node;
(4) If p^t is the super node, calculate p^* and p^t corresponding values in the super node table $Sim(p^*, ID^t) \geq \theta$ (t is the number of super node table). If established, connect p^* with the corresponding super nodes and join in the interest domain of corresponding super nodes, then perform step (5), or turn to (6);
(5) After node p satisfied the condition and joined in an interest domain, calculate the similarity s_i of each interest cluster in interest domain

$$Sim(p, IC_i) = s_i \tag{8}$$

p is the node expression of the node, IC_i is the representation of i-th interest cluster in the list of interest cluster. Rank the value of similarity in descending order, and add node p into the high similarity of former $k(k \leq n)$ interest clusters, n expresses the list size of interest cluster;
(6) p^* constantly discovers new $n(p^*)$ by p^t, namely new node, repeat the process4.

3.3 Exits Interest Domain

In SNS learning community, nodes may add or delete documents according to their needs, which leads to themes change. Once their own themes are not similar to the topics of interest domain, the nodes will exits the interest domain, and delete nodes in list which are not similar to their own interest. When the original node $p^\sharp$ in the interest domain wants to communicate node p^*, it will find the node p^* has retired, then node $p^\sharp$ will update its own repository.

4 Interest Domain Searching Mechanism in SNS Community

When node q searches, the search request is sent to all the super nodes in local interest cluster at first, then is proceed within this interest cluster.While the local super node will also send the same search request to some of the super nodes in neighbor interest clusters so that request can be proceed in other cluster.

4.1 Searching within Interest Clusters

The super node p_j, which has received the query request, searches through document sets of local nodes. Meanwhile, it will also select an appropriate amount of leaf nodes in the interest cluster to complete the searching task. The process is as follows:

(1) For entry t_n , use the language model $p(t_n|M_{d_k})$ to calculate the frequency of t_n in the document set C_j:

$$f_n = \frac{\sum_{d \epsilon C_j} p(t_n|M_{d_k})}{|C_j|_d} \tag{9}$$

Among them,$|C_j|_d$ is the number of documents in document set $|C_j|$ of super node p_j.

(2) Through the introduction of appropriate Kullback-Leibler[8,9](K-L) divergence algorithm calculates the interest similarity of computational node q and document set C_j:

$$rel(q, C_j) = KL(q, C_j) = \sum_{t_n \in q} p(t_n|q)| \log \frac{p(t_n|q)}{p(t_n|C_j)}| \tag{10}$$

Put the calculated degree of interest correlation into the descending order and choose the first K leaf nodes to perform searches.

4.2 Searching between Interest Clusters

Super node p_j will forward searching request to other super nodes in the same interest domain, which need to be appropriate selected from the neighbor cluster to perform searching, while searching in the local interest cluster. The process is as follows:

(1) The first super node p_j in interest domain receives the requesting from node q and set a value of TTL[10] (Time-to-Live) for q. Each time the interest cluster searching performs, TTL will be corresponding reduced 1, and when TTL=0, turn to search within the domain interest.

(2) Neighbor super node p_i receives the forwarded searching, then calculates its interest vector $\boldsymbol{V_{sij} = (v_{sij1}, v_{sij2}, \cdots v_{sijn})}$, the $\boldsymbol{V_{si} = (v_{si1}, v_{si2}, \cdots v_{sin})}$ of node q, and the interest correlation between the two nodes:

$$Sim(\boldsymbol{V_{sij}}, \boldsymbol{V_{si}}) = \frac{\sum_{k=1}^{n} |v_{si1k} - \bar{v_{sik}}| \cdot |v_{si2k} - \bar{v_{sik}}|}{\sqrt{\sum_{k=1}^{n} (v_{sijk} - \bar{v_{sijk}})^2} \cdot \sqrt{\sum_{k=1}^{n} (v_{sik} - \bar{v_{sik}})^2}} \tag{11}$$

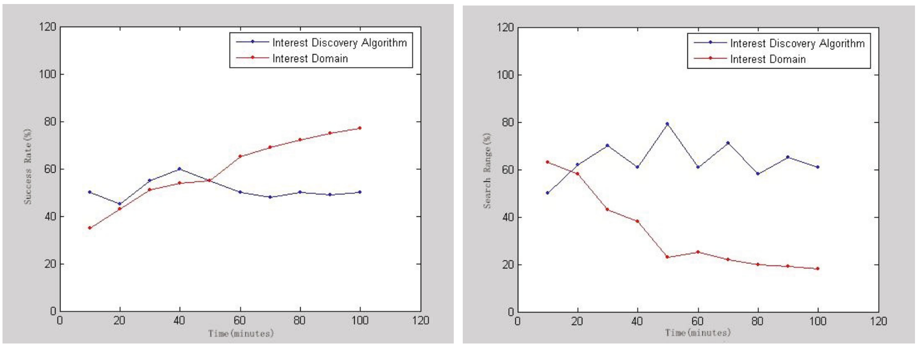
send the result to p_i .

(3) After p_i receives the correlation interest values of neighbor super nodes which are calculated above, put these nodes in descending order according to their values, selects the top $\delta_2\%$, then make these nodes performs the searching task. Certainly, super node that performs the searching mission will decrement the TTL by one, and then forwards the searching request of q, after that searching is performed within each interest cluster.

(4) If it is known in advance that which interest cluster the object resources are in, then the above-mentioned approach of searching will be used within interest cluster.

5 Simulation Experiment and Performance Evaluation

In the simulation experiment environment, constructing a SNS learning community peer network with 1000 nodes, and each node was denoted by a specific data structure. At the beginning, nodes connect to other nodes randomlyand the average degree are 3. Searching request and response between nodes are passed by parameters. Experimental data comes from the date set of Domain, Document and Topic information in Routing Task of TREC8[11,12,13]. Figures.(a) and Figures.(b) of Fig.2 respectively show comparison of search success rate and comparison of average search scope between existing interest discovery algorithm and searching mechanism proposed in this paper



(a) Comparison of search success rate **(b)** Comparison of average search scope

Fig. 2. Respectively shows comparison of search success rate and comparison of average search scope between existing interest discovery algorithm and searching mechanism proposed in this paper

It can be seen from the comparison of Figures (a) and (b), after the formation of the effective interest domain, the interest searching mechanism had a high searching success rate and the average search volume is greatly reduced.

6 Conclusion

This paper presents an interest-based searching mechanism,which denotes the interest of learners in greater detailand considers computational complexity and comprehensiveness of factors to calculate the similarity of interests.This interest searching mechanism forms the nodes whose interest similarity exceeds a certain value to an effective interest domainand conducts interest searching within interest cluster or interest domain. The experimental results show that, the searching mechanism has higher success rate, more stable , and the average searching scope is smaller.

References

1. Doraisamy, R., Radhakrishnan, S., et al.: The effectiveness of integrated teaching over traditional teaching among first year mbbs students: A preliminary study. Medical Journal of Dr. DY Patil University 6, 139 (2013)
2. Luo, J., Gu, W.X.: Establishment of network platform of virtuai teaching laboratories in colleges and universities based on jsp technology. In: Advanced Engineering Forum, vol. 4, pp. 189–192. Trans. Tech. Publ. (2012)
3. Ring, M.: Integrating facebook into distance education and online learning environments: To promote interactive online learning communities (2012)
4. Tagawa, T., Yamakawa, O., Yasutake, K., Sumiya, T., Inoue, H.: Finding characteristic part of interaction inside sns as the learning community. In: Society for Information Technology & Teacher Education International Conference, vol. 2012, pp. 3791–3795 (2012)
5. Yuan, T.: Research on construction based on sns learning community. Gakuen: Education and Scientific Research, 40 (2012)
6. Li., G.: Reflection on establishment of sns learning community in campus network. Journal of Nanchang College of Education (2012)
7. Zhenchao, Z., Yaoping, F., Li, M.: Node clustering algorithm based on network coordinate for unstructured p2p. Computer Project 36, 98–100 (2010)
8. Hershey, J.R., Olsen, P.A.: Approximating the kullback leibler divergence between gaussian mixture models. In: IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. 317–320 (2007)
9. Pinto, D., Benedí, J.-M., Rosso, P.: Clustering narrow-domain short texts by using the kullback-leibler distance. In: Gelbukh, A. (ed.) CICLing 2007. LNCS, vol. 4394, pp. 611–622. Springer, Heidelberg (2007)
10. Tang, X., Xu, J., Lee, W.-C.: Analysis of ttl-based consistency in unstructured peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems 19, 1683–1694 (2008)
11. Prager, J., Radev, D., Brown, E., Coden, A., Samn, V.: The use of predictive annotation for question answering in trec8. Information Retrieval 1, 4 (1999)
12. Yilmaz, E., Aslam, J.A., Robertson, S.: A new rank correlation coefficient for information retrieval. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 587–594. ACM (2008)
13. Webber, W., Moffat, A., Zobel, J., Sakai, T.: Precision-at-ten considered redundant. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 695–696. ACM (2008)

Improving the Frequency Adaptive Capability of Hybrid Immune Detector Maturation Algorithm

Jungan Chen, ShaoZhong Zhang, and Danjiang Chen

Electronic Information Department
Zhejiang Wanli University
No.8 South Qian Hu Road
Ningbo, Zhejiang, 315100, China
friendcen21@hotmail.com, dlut_z88@163.com, chendj@zwu.edu.cn

Abstract. In abnormal detection, the frequency of abnormal activities is changed over the time, so it is reasonable that detection algorithms can be adapted to the frequency's change. In this work, an adaptive mathematic model is proposed to improve the adaptive capability. Then an augmented hybrid immune detector maturation algorithm applied in anomaly detection is presented. Experiment results show the algorithm can be adapted to the frequency's change.

Keywords: Artificial immune system, adaptive capability, hybrid immune detector.

1 Introduction

Nowadays, Artificial Immune System (AIS) has been applied to many areas such as computer security, classification, learning and optimization [1]. Negative Selection Algorithm, Clonal Selection Algorithm, Immune Network Algorithm and Danger Theory Algorithm are the main algorithms in AIS [2,3].

In abnormal detection, the frequency of abnormal activities is always changed over the time. So it is required that the detection algorithms can be adapted to the frequency's change. When abnormal attack's frequency increased intensely, the number of these detectors generated in AIS algorithm must be increased rapidly. Otherwise, abnormal attacks would be fail because fewer detectors are generated to detect these abnormal.

In our recent works, inspired from reference [4], Hybrid Immune Detector Maturation Algorithm (HIDMA) is proposed to combine TMA with affinity maturation and solves the population-adapt problem[5]. Lifecycle Model in HIDMA-LM is proposed, where the detectors can be adapted to the nonself(abnormal)'s change [6]. To improve the generalization capability [7],HIDMA-GC (Generalization Capability) is proposed[8].

In this work, a frequency adaptive mathematic model is proposed to improve the adaptive capability, so the number of detectors generated can be adapted to the change of the abnormal activities and its frequency. At last the algorithm HIDMA-FA(Frequency Adaptive) is proposed.

2 Related Work

2.1 Match Range Model

U={0,1}^{n} ,n is the length of binary string. The normal set is defined as selves and anomaly set is defined as nonselves. selves∪nonselves=U. selves∩nonselves= Φ. There are two binary strings Sg=g$_1$g$_2$...g$_n$, Sb=b$_1$b$_2$...b$_n$. The hamming distance between Sg and Sb is:

$$\mathrm{d(Sg,Sb)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \overline{\mathrm{g_i} \oplus \mathrm{b_i}} \tag{1}$$

The detector is defined as dct = {<Sb, selfmin, selfmax>|Sb∈U, selfmin, selfmax∈N}. selfmax is the maximized distance between dct.Sb and selves , selfmin is the minimized distance. The detector set is defined as DCTS. Selfmax and selfmin is calculated by setMatchRange(dct, selves), i∈[1, |selves|], self$_i$ ∈selves

$$setMatchRange = \begin{cases} selfmin = min(d(self_i, dct.sb)) \\ selfmax = max(d(self_i, dct.sb)) \end{cases} \tag{2}$$

[selfmin,selfmax] is defined as self area. Others are as nonself area. The antigen is defined as Ag= {<Sg >|Sg∈U}, The antigen set is defined as AGS.

Suppose there is one antigen ag∈AGS and one detector dct ∈ DCTS. When d(ag.Sg,dct.Sb) ∉[dct.selfmin, dct.selfmax], ag is detected as anomaly. It is called as Range Match Rule (RMR) shown in equation3.Value true means that ag is anomaly.

$$RMRMatch(ag, dct) = \begin{cases} false, d(ag.sg, dct.sb) \in [dct.selfmin, dct.selfmax] \\ true, d(ag.sg, dct.sb) \notin [dct.selfmin, dct.selfmax] \end{cases} \tag{3}$$

Based on RMR, the detect procedure detect(ag,DCTS) is defined as equation4. True means that ag is anomaly.

$$detect(ag, DCTS) = \begin{cases} true, \exists(dct_k \in DCTS), RMRMatch(ag, dct_k) = true \\ false, others \end{cases} \tag{4}$$

2.2 The State Transformation Model

In this model, the antigen is redefined as Ag= {<Sg,state,undetectedCount >|undetectedCount∈N, state∈{'new','suspect','self','nonself'}}.The detector is redefined as dct = {<Sb, d, harmmax, selfmin, selfmax,state,lifecycle, detectedAgNum, oldDetectedAgNum >|d,harmmax, selfmin, selfmax,lifecycle, detectedAgNum, oldDetectedAgNum∈N,state∈{'new','highest','maturation','die'}}.

Other properties in the definition of Ag, dct are calculated by the following steps:

$$M = |AGS|, N = |DCTS|, i \in [1, m], j \in [1, N] \tag{5}$$

The value i is the index of antigen in AGS and the value j is the index of detector in DCTS. The value of d is the distance between dct and current antigen Ag.

In equation 6, If detector y detects the antige i, antigen i is changed to nonself antigen and detector y is changed to maturation detector. The detectedAgNum of detector y is increased . $\mathrm{DCTS_{iM}}$ is defined as the set of detectors which can detect the antigen i as nonself.

$$\begin{array}{l}\mathrm{if(RMRMatch(Ag_i,dct_y))} \\ \begin{cases}\mathrm{Ag_i.state = 'nonself'} \\ \mathrm{dct_y.state = 'maturation'} \\ \mathrm{dct_y.detectedAgNum = dct_y.detectedAgNum + 1} \\ \mathrm{DCTS_{iM} = DCTS_{iM} \cup dct_y}\end{cases}\end{array} \tag{6}$$

In equation 7, $\mathrm{r_{life}}$ is a parameter used to control the lifecycle of maturation detector. Suppose detector $\mathrm{dct_m}$ has the max detectedAgNum, $\mathrm{dct_m}$'s lifecycle is increased after $\mathrm{ag_i}$ is detected. So $\mathrm{dct_m}$ can be reserved to detect more similar antigens and the generalization capability of the algorithm is improved. If $\mathrm{r_{life}}$ is set to ∞, it will not die.

$$\begin{array}{l}\exists \mathrm{dct_m} \in \mathrm{DCTS_{iM}} \\ \mathrm{dct_m.detectedAgNum = max(dct_*.detectedAgNum)} \\ \mathrm{AgNum = dct_m.detectedAgNum - dct_m.oldDetectedAgNum} \\ \mathrm{if(AgNum > 0)\{} \\ \mathrm{dct_m.lifecycle = dct_m.lifecycle + r_{life}{}^*AgNum} \\ \mathrm{dct_m.oldDetectedAgNum = dct_m.detectedAgNum} \\ \}\end{array} \tag{7}$$

Other detail is required to reference to[8].

3 Frequency Adaptive Model

W_{gen} is defined as the sample window's size. g is the generaton's value.N_g is defined as the number of abnormal detected in generation g. α and β are the parameters used to control the lifecycle of one detector. In this paper, the equation 7 is changed as the equation 9.

$$F_g = N_g/W_{gen} \tag{8}$$

$$\begin{array}{l}\exists \mathrm{dct_m} \in \mathrm{DCTS_{iM}} \\ \mathrm{dct_m.detectedAgNum = max(dct_*.detectedAgNum)} \\ \mathrm{AgNum = dct_m.detectedAgNum - dct_m.oldDetectedAgNum} \\ \mathrm{if(AgNum > 0)\{} \\ \mathrm{dct_m.lifecycle = dct_m.lifecycle} + \alpha^*\mathrm{AgNum} + \beta * F_g \\ \mathrm{dct_m.oldDetectedAgNum = dct_m.detectedAgNum} \\ \}\end{array} \tag{9}$$

AgNum is the number of abnormal detected in current generation g. The detector with bigger AgNum can be reserved to detect more similar antigens and the generalization capability of the algorithm is improved. Furthermore, the bigger the frequency of abnormal activities is, the longer the winner in the detectors population can live.

4 Experiments

The objective of the experiments is to investigate the frequency adaptive capability. Experiments are carried out using the famous benchmark Fisher's Iris Data. Minimal entropy discretization algorithm is used to discretize these data sets [9].

Table 1. The value of the parameters

Parameters	Values
W_{gen}	5,10,20,40,100
α	0,5,10,15,20,30,40
β	0,5,10,15,20,30,40

For verifying the adaptive character, nonself data are changed every 20 generations, maxundetectCount=maxg. In the Iris Data, It has 4 attributes and has total 150 examples with three classes: 'Setosa', ' Versicolour', ' Virginica'. Each class has 50 examples. ' Virginica' is considered as normal data. The other two are considered anomaly and injected into the algorithm in turn and repeatedly. The proposed algorithm HIDMA-FA runs for 10 times especially with different W_{gen},α,β which are list int table 1. The max generation maxg=1000000.

4.1 Frequency's Curve

In Fig.1 , it is shown that the value of Frequency can reflect the change of nonself. The smaller the value of W_{gen} is, the bigger the change range is. The frequency's value Fg is reflected more precisely with the change of nonself at W_{gen}=5, So W_{gen}=5 is choose as a based condition in the following discussion.

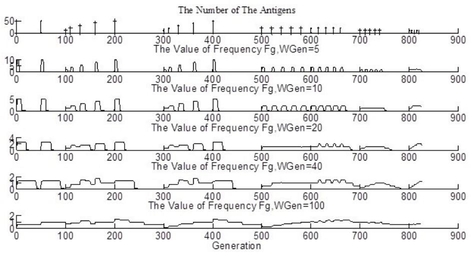
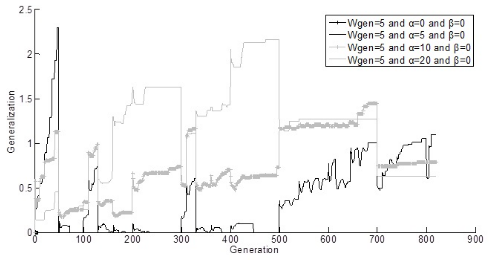
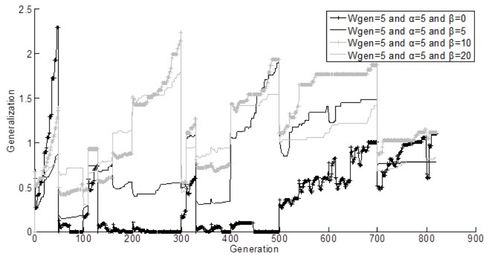


Fig. 1. The frequency's curve using Iris Data

4.2 Comparison of Generalization



(a)



(b)

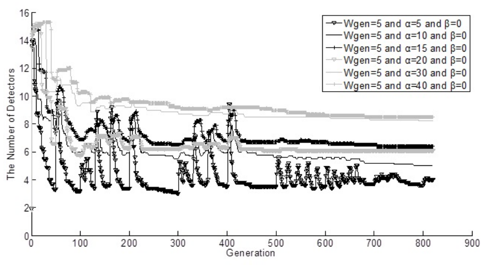
Fig. 2. Comparing with HIDMA-GC

In Fig.2 , when β =0, the algorithm is similar with HIDMA-GC. So it is shown that α can regulate the generalization capability in HIDMA-GC in Fig.2(a). Similarly, the Generalization's value is increased with β in most case in Fig.2(b), so HIDMA-FA(β>0) has more generalization capability than HIDMA-GC(β =0).

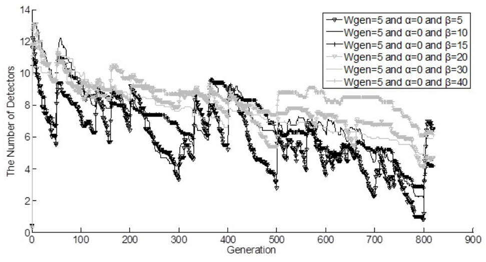
But in β =20, the generalization's value is smaller than others sometimes especially when generation is larger than 500. It is because that the valid detector is easy to life for longer when the change interval of nonself's number is less than 20 and there is no life pressure to force the detectors to improve their generalization capability. In a word, HIDMA-GC can improve the generalization capability through the Frequency Adaptive Model.

4.3 The Effect of α and β

As the bigger α and β are, longer the lifecycle of the detectors are, so the number of detectors become more and more with α,β increasing in Fig.3. The smoothness of the curve reflects the stability of the detector population.



(a)



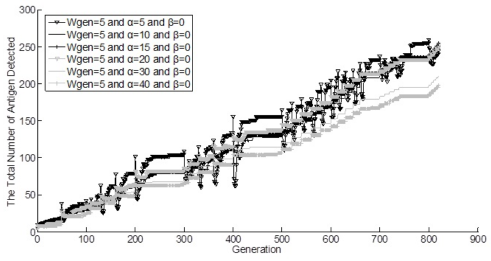
(b)

Fig. 3. The number of detectors

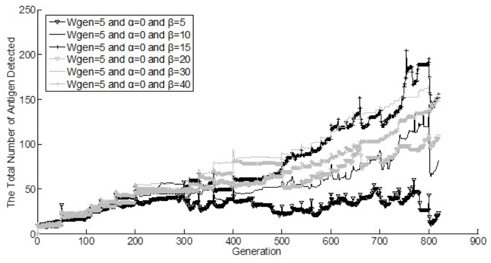
In Fig.3(a), α can regulate the number of detectors and improve the generalization capability of the detectors. In Fig.3(b), it is shown that the number of detectors is sensitive with the change of nonself shown in Fig.1 and the changed frequency of the number of detectors is different with β changing. So β can regulate the degree of detector population's response to the nonself's change. In a word, α and β have different function, one for population capability, generalization capability and one for frequency adaptive capability.

As β can regulate the degree of detector population's response to the nonself 's change, the total number of antigen detected by the detector population is sensitive with β in Fig.4(b).

The stability of the detector population will influence the agent's detection. It is discussed that α can regulate the number of detectors and improve the generalization capability of the detectors. When α is increasing, the detector population's stability is improved in Fig.3(a). So the average number of antigens detected by all detectors is sensitive with α in Fig.5(a).

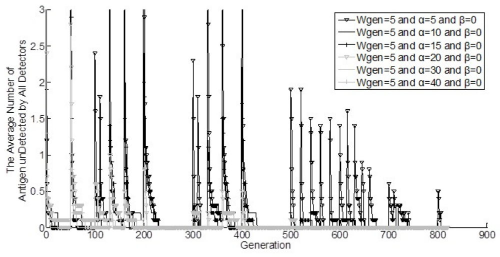


(a)



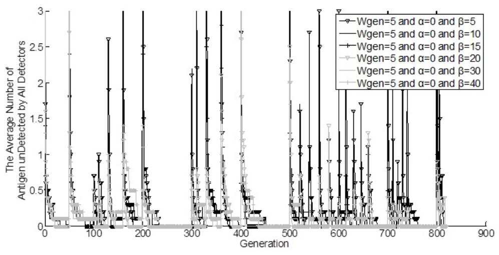
(b)

Fig. 4. The total number of antigens detected



(a)

Fig. 5. The average number of agentigens undetected



(b)

Fig. 5. (*Continued*)

5 Conclusion

In this work, to improve the frequency adaptive capability, an augmented HIDMA algorithm (HIDMA-FA) is proposed. The results show that the frequency adaptive capability is improved. It is concluded that HIDMA-FA has the different regulation functions through α and β. α is used for the population capability, generalization capability. β is used for the frequency adaptive capability. Furthermore, the optimized α and β are required to be solved in the future.

Acknowledgement. This work is supported by National Natural Science Foundation of China 71071145, Zhejiang Public Welfare Technology Research Project Foundation 2013C31122, Zhejiang Provincial Nature Science Foundation Y1110200 ,Ningbo Nature Science Foundatio2012A610007, 2013A610070, The Science and Technology Innovation Team of Ningbo (Grant No.2013B82009, 2012B82006).Thanks for the assistance received by using KDD Cup 1999 data set [`http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`]

References

1. Hart, E., Timmis, J.: Application areas of AIS: The past, the present and the future. Journal of Applied Soft Computing 8, 191–201 (2008)
2. Timmis, J., et al.: An interdisciplinary perspective on artificial immune systems. Evolutionary Intelligence 1, 5–26 (2008)
3. Greensmith, J., Aickelin, U., et al.: Information Fusion for Anomaly Detection with the Dendritic Cell Algorithm. Information Fusion 11, 21–34 (2010)
4. Hofmeyr, S.: A.: An Immunological Model of Distributed Detection and its Application to Computer Security.PhD Dissertation. University of New Mexico (1999)
5. Chen, J., Chen, W., Liang, F.: Adaptive hybrid immune detector maturation algorithm. In: Corchado, E., Graña Romay, M., Manhaes Savio, A. (eds.) HAIS 2010, Part II. LNCS, vol. 6077, pp. 201–208. Springer, Heidelberg (2010)

6. Chen, J., et al.: Hybrid Immune Detector Maturation Algorithm with LifeCycle Model. In: International Conference on Computer Science and its Applications, pp. 213–218. IEEE Press, New York (2009)
7. de Castro, L.N., et al.: Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation 6, 239–251 (2002)
8. Chen, J., Liang, F., Fang, Z.: Improving the Generalization Capability of Hybrid Immune Detector Maturation Algorithm. In: Corchado, E., Snášel, V., Abraham, A., Woźniak, M., Graña, M., Cho, S.-B. (eds.) HAIS 2012, Part III. LNCS, vol. 7208, pp. 298–308. Springer, Heidelberg (2012)
9. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features, `http://robotics.stanford.edu/~ronnyk/disc.ps`

Cluster-Based Time Synchronization Protocol for Wireless Sensor Networks*

Jian Zhang, Shiping Lin, and Dandan Liu

Computer School,
WuHan University,
WuHan, China
{jzhang,csliudd}@whu.edu.cn, lsp2046@126.com

Abstract. Time synchronization is one of the most important premises for many applications in wireless sensor networks, including data fusion, target tracking, military surveillance, etc. Existing protocols generally suffered from higher energy consumption, worse accuracy or applicable only in small wireless sensor networks. In this paper, we propose a novel cluster based time synchronization method for large-scale wireless sensor networks. First, a cluster tree is constructed as the backbone. Then, after network clustering is completed, different time synchronization mechanisms are applied within cluster head nodes and cluster members, respectively. The theoretical analysis and simulation results show that the proposed algorithm can effectively reduce energy consumption comparing to existing protocols, while it also achieves the high accuracy of time synchronization.

Keywords: Time synchronization, clustering, energy efficient, wireless sensor networks.

1 Introduction

Time synchronization is one of the most important premises for many applications in wireless sensor networks (WSN), such as data fusion, node localization, target tracking, military surveillance, and so on. Because of the large number of sensor nodes, WSN generally is limited in the nodes' energy, network communication bandwidth, processing power, etc. Therefore, the design and implementation of time synchronization in WSN must consider much about good scalability, low energy consumption and high reliability, etc. [1]. On the other hand, due to the inherent characteristics of the hardware, the internal clock of node always has the cumulative deviations over time. Thus, the time synchronization aims at eliminating this deviation, and reaching the time synchronization of the entire network. Therefore, due to those limitations, it is a big challenge to explore an energy efficient time synchronization mechanism for WSN.

* This work was supported by the National Natural Science Foundation of China (No. 61103216), the Natural Science Foundation of HuBei Province of China (No. 2011CDB458), the Natural Science Foundation of HuBei Province of China (No.2013CFB295), and the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20110141120038).

Many time synchronization protocols and approaches [2] have been proposed by research scholars and institutions recently. Broadcast is the most direct way to achieve synchronization among nodes receiving the message. However, it can only be achieved among nodes within the transmission radius, and the energy consumption for time synchronization is relatively high. Moreover, sometimes the root for broadcasting may become the most vulnerable point of the network. M. Maróti et al proposed to mark time stamp in the MAC layer for synchronization. But it required more support from hardware, thus is not suitable for general circumstances. Hierarchical or tree-based methods are effective ways for applications in large-scale networks. Existing such methods suffered from higher computation complexity, or lower synchronization accuracy in exchange for complexity reduction.

To avoid the drawbacks of existing approaches stated above, we proposed a Cluster-based Time Synchronization (CTS) protocol for large scale wireless sensor networks. The networks are divided into several clusters first. All the cluster heads and the root are constructed as a tree and act as the backbone for the network. Then different time synchronization mechanisms are applied within the cluster heads tree and inside each cluster respectively, so that the overall energy consumption can be significantly reduced. The performance of the proposed CTS mechanism is verified through theoretical analysis and simulations. We also compare our CTS protocol with existing TPSN protocol. As shown from the simulation results, CTS achieves higher energy efficiency by significantly reducing the entire communication overhead without increasing much on the synchronization deviation.

The paper is organized as follow. Section II introduces the related works in time synchronization. We illustrate the proposed protocol in detail and corresponding theoretical analysis in Section III. The performance evaluation is given in section IV. Finally we conclude this paper in Section V.

2 Related Work

The problem of time synchronization in WSN was proposed by Jeremy Elson and Kay Romer et al. in 2002 HotNets[2] conference. This concept has a far-reaching impact on wireless sensor networks. Later on, researches on time synchronization for wireless sensor networks began to flourish. Research scholars and institutions have proposed many time synchronization protocols [2], such as RBS[3], TPSN[4], DMTS[5], FTSP[6], LTS[7], SLTP[8] and so on.

RBS (Reference Broadcast Synchronization) [3] protocol is one of the most classic time synchronization protocol, based on the idea of third-party node making time synchronization. In this algorithm, nodes broadcast messages to their neighboring nodes. These neighboring nodes exchange the information about their receiving time, and calculate the time deviation value so as to achieve time synchronization within this group. The main limitation of RBS is the time synchronization can only be achieved among nodes within the transmission radius, and the energy consumption for time synchronization is relatively high.

TPSN (Timing-Sync Protocol for Sensor Networks) [4] protocol is a time synchronization algorithm that suitable for the whole wireless sensor networks. The synchronization algorithm has two-stage processes: network classification and time synchronization. First, the network is divided into several levels. There is only one node at level 0 and is called the root node. In the time synchronization phase, the i-th-level node has bidirectional synchronization with the (i-1)-th-level node, and then the (i+1)-th-level node has bidirectional synchronization with the i-th-level node. This process is repeated until all nodes reach the synchronization. The synchronization accuracy of TPSN is higher than that of RBS, and the scalability of TPSN is better as well. However, as the network has only one root node, it has higher probability to bring about single point of failure.

DMTS (Delay Measurement Time Synchronization) [5] protocol realizes time synchronization among nodes by estimating and measuring the unidirectional transmission delay. Compared with the synchronization protocols stated above, the communication overhead of DMTS is smaller, the computation complexity is lower, but its synchronization accuracy is worse than RBS and TPSN.

FTSP (Flooding Time Synchronization Protocol) [6] protocol estimates bit offset by marking time stamp in the MAC layer and using the linear regression method. It reduces the probability of delay, and improves synchronization accuracy. FTSP also takes many factors into account, such as the root node selection, link failure, the root node failure, the topology changes, redundant information and the problem of multiply root nodes. It has strong robustness, but the versatility is poor. It needs more hardware to support it, and the energy consumption is relatively high.

LTS (Lightweight Tree-Based Synchronization) [7] protocol is a time synchronization protocol based on spanning tree. To reduce the protocol complexity, it decreased the requirements for the synchronization accuracy. Therefore, the time synchronization accuracy is relatively low. Meanwhile, its synchronization accuracy is associated with the depth of the network spanning tree. The greater the depth, the lower the accuracy.

SLTP (Scalable Lightweight Time Synchronization) [8] protocol is a cluster-based time synchronization protocol. It includes configuration phase and the synchronization phase. It has static mode and dynamic mode respectively, and therefore is adaptable to both static and dynamic networks. But the computation process is quite complicated.

The research on time synchronization in WSN starts about ten years ago, but it has already produced a lot of research achievements. However, some of the protocols and algorithms stated above only take into account the small-scale networks, and are not suitable for the large-scale networks. Some other cluster-based protocols can be applied to the large-scale network, but the corresponding energy consumption is too high. So it is necessary to explore an energy-efficient time synchronization protocol that is applicable to a large-scale network.

3 Time Synchronization

We proposed a novel cluster-based time synchronization protocol for wireless sensor networks. The proposed protocol is fully distributed, based on sensor collaboration. Before going to the specification of the protocol, we first introduced the network model and some assumptions that should follow.

3.1 Network Model

The network is first divided into several clusters by MLC algorithm [11]. Clustered network model is shown in Figure 1. Sensors should follow the specific conditions stated below [9]:

- Sensors are homogeneous and have the same transmission radius, except the sink node [10].
- Sensors are randomly deployed in the monitoring area, and remain static after their deployment.
- All sensor nodes in the network have a unique ID.

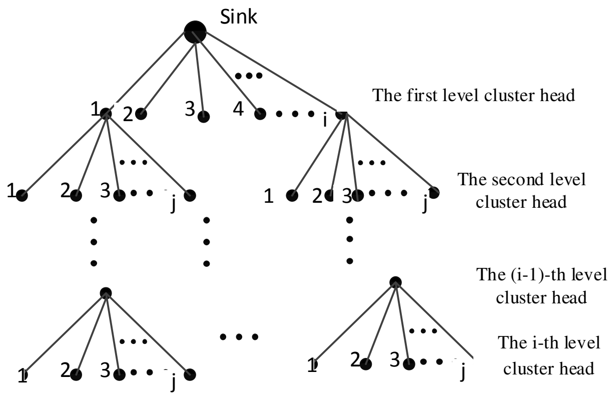

Sink
The first level cluster head
The second level cluster head
The (i-1)-th level cluster head
The i-th level cluster head
1
2
3
4
i
j

Fig. 1. Clustering Network Architecture

3.2 Time Synchronization between Cluster Heads

The sink node and all cluster head nodes together are considered as the backbone for the network. A bidirectional protocol similar to TPSN can be used to achieve time synchronization among backbone nodes [12].

Here, we explain the details about time synchronization between two nodes. As shown in Figure 2, T1 and T4 represent the local time of node A, T2 and T3 represent the local time of node B. At time T1, the node A sends a time synchronization packet to node B. The time synchronization packet contains the information of node A's level and the value of T1. Node B receives the synchronization packet at time T2.

Here T2=T1 + Δ + d, where Δ represents the time deviation between A and B and d represents the transmission delay from A to B. At time T3, the node B sends an acknowledgment packet to the node A. The acknowledgment packet contains the information of node B's level and the values of T1, T2, T3. Then node A receives the packet at time T4. Assume that the time deviation and transmission delay is constant, then node A can calculate the numbers.

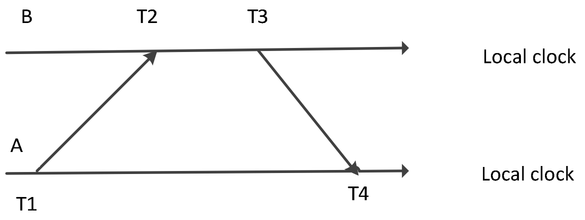


Fig. 2. Bidirectional Time Synchronization between Two-level Nodes

Specifically, according to the data transmission process between nodes A and B (Fig.2), we can get the following two equations:

$$T2=T1+\Delta+d \tag{1}$$

$$T4=T3-\Delta+d \tag{2}$$

From the above two equations, we have:

$$\begin{cases} \Delta = \dfrac{(T2-T1)-(T4-T3)}{2} \\ d = \dfrac{(T2-T1)+(T4-T3)}{2} \end{cases} \tag{3}$$

After time deviation Δ is calculated, node A is able to achieve time synchronization with the reference node B by adjusting with the deviation value.

Based on the time deviation obtained, the time synchronization process will be operated as follow. First, the sink node starts time synchronization by broadcasting a time synchronization package to all cluster heads in 1st-level. After receiving the package, all cluster head nodes in the 1st-level wait for a random amount of time to go on bidirectional message exchange process with the sink node. Waiting a random time can help to avoid collisions on the data link layer. After receiving the response message, the cluster head nodes adjust their time to achieve time synchronization with the sink node. Similarly, the cluster head nodes in the 2nd-level also overhear messages during the exchange process. When they receive the message, the nodes will wait for a random time, and then begin the bidirectional message exchange process [13] with the 1st-level head nodes.

This process is repeated until all cluster head nodes of the backbone finish time synchronization with the sink node.

3.3 Time Synchronization between Cluster Head and Members

The number of nodes in the cluster is much larger than the number of cluster head nodes. In order to reduce energy consumption of time synchronization for cluster members, we adopt a unidirectional synchronization broadcasting mechanism by cluster head nodes to perform time synchronization among nodes in the cluster.

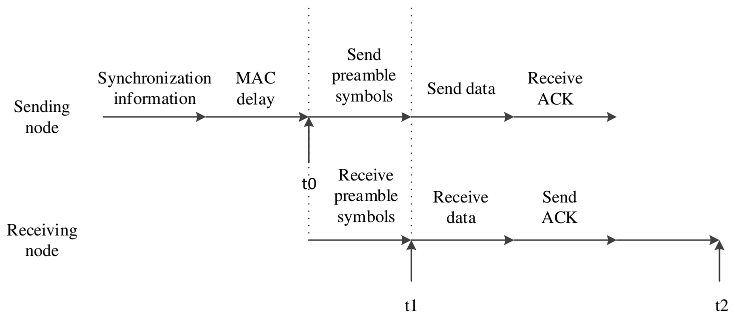


Fig. 3. Time synchronization process between head and members

After the i-th-level cluster heads complete time synchronization with the (i +1)-th-level cluster heads, the i-th level cluster head nodes will start time synchronization process within the corresponding cluster. Synchronous message exchange process is shown in Figure 3. The cluster heads keep monitoring the communication channel with each cluster members. When the channel became idle, they generate a synchronization packet with timestamp marked as t0. Note that before sending the synchronization packet, cluster head nodes will send the preamble symbols, in order to synchronize with the receiving node. Based on the length of preamble symbols n and the forwarding time t per bit, we can estimate the transmission time nt for the entire preamble symbols. When synchronization packets arrive, receiving nodes mark the timestamp as t1, adjust their local clock recorded as time t2 before and change it to $t0 + nt + (t2 - t1)$.

When all sensor nodes in the network reached a time synchronization state, the operation is completed.

3.4 Deviation Analysis

As we use different time synchronization mechanisms for cluster head nodes and cluster members respectively, the synchronized deviation analysis for the proposed protocol should be divided into two parts [14] as well.

3.4.1 Deviation Analysis about Time Synchronization between Cluster Heads

Consider the similar situation in the message transmission process in Figure 2, where node A sends a message packet to node B. The time at each node according to internal clock is represented by uppercase letters (i.e., T1, T2), while the external standard time is denoted by lowercase letters (i.e., t1, t2). We can obtain the following equations:

$$\begin{cases} t2 = t1 + S^A + P^{A\to B} + R^B \\ T2 = T1 + S^A + P^{A\to B} + R^B + D_{t1}^{A\to B} \end{cases} \tag{4}$$

Here, S^A represents the time for node A to send a packet, $P^{A\to B}$ represents the transmission time between node *A* and node *B*, R^B represents the time for node *B* to receive a packet. All these denotations are based on external standard clock. $D_{t1}^{A\to B}$ represents the clock drift between node *A* and node *B* at time *t1*.

Then node *B* sends a response message at time *T3*, and node *A* receives the message at time *T4*. Thus, we have $T4 = T3 + S^B + P^{B\to A} + R^A + D_{t4}^{A\to B}$, where S^B represents the time for node *B* to send a packet, $P^{B\to A}$ represents the transmission time between node *A* and node *B*, R^A represents the time for node *A* to receive a packet. $D_{t4}^{A\to B}$ represents the clock drift between node *A* and node *B* at time *t4*.

Because $D_{t3}^{B\to A} \approx D_{t4}^{A\to B} = -D_{t4}^{B\to A}$, we divide $D_{t1}^{A\to B}$ into:

$$D_{t1}^{A\to B} = D_{t4}^{A\to B} + RD_{t1\to t4}^{A\to B} \tag{5}$$

where $RD_{t1\to t4}^{A\to B}$ represents the relative clock drift between node A and node B from time t1 to time t4.

And we can obtain the following result:

$$\Delta = \frac{(T2-T1)-(T4-T3)}{2} = \frac{S+P+R+RD_{t1\to t4}^{A\to B} + 2D_{t4}^{A\to B}}{2} \tag{6}$$

where S, R and P represent the deviation of sending delay, receiving delay and transmission delay, respectively. They can be derived from the following formula.

$$S = S^A - S^B \tag{7}$$

$$P = P^{A\to B} - P^{B\to A} \tag{8}$$

$$R = R^B - R^A \tag{9}$$

Note that $D_{t4}^{A\to B}$ is the clock drift between node A and node B at time t4. Thus the deviation of time synchronization will be

$$Deviation = \Delta - D_{t4}^{A\to B} = \frac{S+P+R+RD_{t1\to t4}^{A\to B}}{2} \tag{10}$$

As shown from the results above, the deviation of time synchronization among cluster head nodes has a linear relationship with respect to the deviation of sending delay, transmission delay, receiving delay and the relative clock drift between two nodes.

3.4.2 Deviation Analysis about Time Synchronization between Cluster Head and Members

Similarly with the methodology of denotations in section 1), uppercase letters are used to represent the time at each node according to internal clock (i.e., T0, T1), and lowercase letters are external standard time (i.e., t0, t1). As shown in Figure 3, the unidirectional synchronization process is:

$$T2 = T0 + T^{A\to B} + P^{A\to B} + R^{B} + D_{t0}^{A\to B} \tag{11}$$

$$D_{t0}^{A\to B} = D_{t2}^{A\to B} + RD_{t0\to t2}^{A\to B} \tag{12}$$

Here, $T^{A\to B}$ represents the time for preamble symbols transmission from node A to node B, $P^{A\to B}$ represents the transmission time of a packet between node A and node B, R^{B} represents the time for node B to receive a packet, $D_{t1}^{A\to B}$ represents the clock drift between node A and node B at time t0. And then:

$$T^{A\to B} + R^{B} = nt + T^{dev} + (T2 - T1) + R^{dev} \tag{13}$$

(n represents the length of the preamble symbols, t is the time for each bit, T^{dev} and R^{dev} denote the deviation for transmission and reception, respectively.)

Thus the clock drift within the cluster is:

$$\Delta = P^{A\to B} + D_{t2}^{A\to B} + RD_{t0\to t2}^{A\to B} + R^{dev} + T^{dev} \tag{14}$$

Then the synchronization deviation for cluster members is:

$$Deviation = \Delta - D_{t2}^{A\to B} = P^{A\to B} + RD_{t0\to t2}^{A\to B} + R^{dev} + T^{dev} \tag{15}$$

Note that the synchronization mechanism within each cluster is unidirectional, therefore it is unable to eliminate the deviations during the transmission and reception process.

4 Performance Evaluation

In this section, we will study the performance of the proposed CTS protocol through simulations. We compare CTS with the classical protocol TPSN. The performance of both protocols are measured in two metrics:

a) Synchronization deviation: the average time synchronization deviation of the entire network, i.e., the average time deviation of all nodes with respect to the sink node;

b) Transmission cost: The number of packets transmitted throughout the network during a time synchronization procedure. This metric is to evaluate the energy efficiency of the protocol.

4.1 Simulation Settings

The simulation environment is Ubuntu + NS2.34. Since NS-2 platform currently has no time synchronization module for wireless sensor networks, we conduct extensions and add the synchronization module to the NS-2 platform [15]. NS2 simulators use two approaches to record the results. One is the trace file, which records the simulation data during the process; the other one is NAM. Finally, we get the results [16] through analyzing trace files.

Record that a cluster heads tree is first constructed as the backbone for synchronization. We denote hierarchy (X-axis in Fig.4) as the levels of the cluster heads tree. It means the member of the x-th level cluster (the cluster head is on the x-th level) is x+1 hops away from the root. We have two sets of experiments. In the first set, the number of nodes is fixed at 200. The cluster heads tree hierarchy is increased from 0 to 7. The objective is to see the variation of synchronization deviation with respective to the hierarchy. In the second set, the nodes number is varied from 150 to 400, to see the average transmission cost during each round of time synchronization. Mobile nodes are randomly deployed in a two dimensional 200×300 network area. Nodes use the omni-directional antenna. The transmission range is fixed at 30m. Each point drawn in the diagrams below is an average of 30 trials, with 95% confidence intervals.

In accordance with previous experimental parameters, we write TCL script, which also includes the definition of component type, delay model, interface queues and other detailed parameters. After we finish Tcl scripts, network simulation experiments can be carried out. Finally, from the generated trace files, we obtain the following results [17].

4.2 Analysis

Figure 4 shows the accuracy of time synchronization vs the hierarchy of cluster heads tree. As shown from the figure, the synchronization deviation of both protocols increase as hierarchy rises up. This is consistent with the intuition. Each cluster head of i-th level is synchronized with the cluster head of i-1-th level. Due to the inherent characteristics of the hardware, the internal clock of node always has the cumulative deviations over time. Thus after several rounds of synchronization from the top to the leaf of the tree, there will be larger and larger deviation.

The synchronization deviation of CTS is slightly larger than that of TPSN. Note that TPSN uses the bidirectional synchronization mechanism. Differently, in CTS protocol, only nodes on the backbone (i.e., the cluster heads) use the bidirectional synchronization, while the cluster members apply the unidirectional synchronization mechanism. The deviation caused by different cluster may accumulated as well. As the number of the cluster members is much larger than that of cluster heads, the synchronization deviation of members occupies a larger proportion of the results. However, we can see from the figure that the increasing ratio of CTS is not high, which can meet the accuracy requirements of wireless sensor network in most applications.

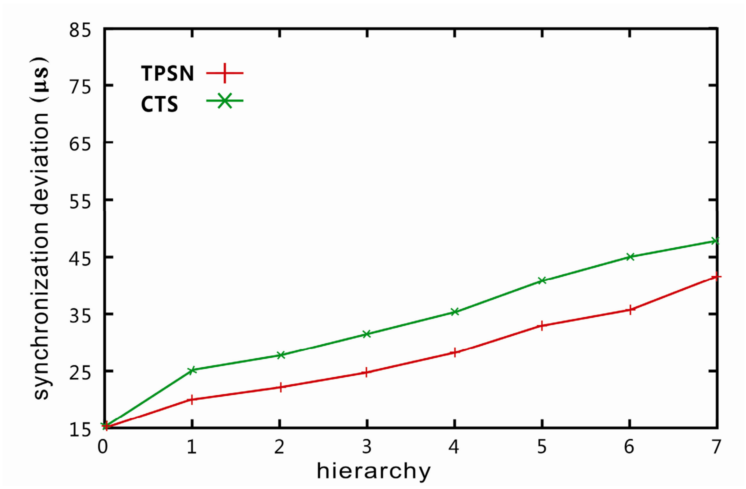


Fig. 4. Relationship between hops and synchronization deviation

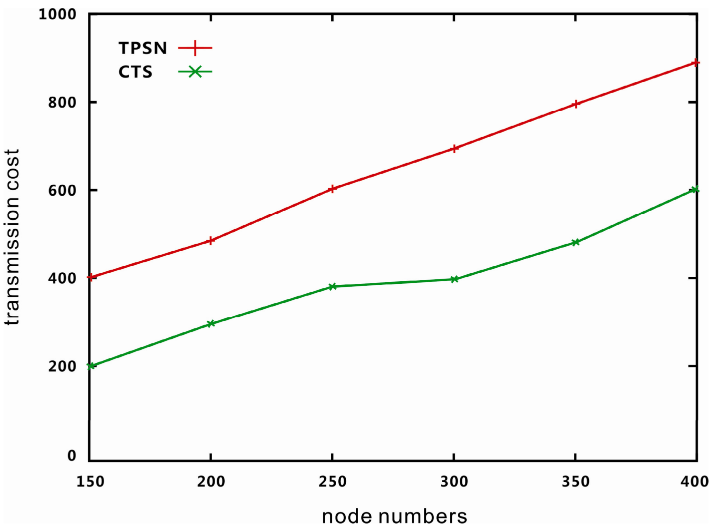


Fig. 5. The relationship between node numbers and message numbers

Figure 5 shows the relationship between transmission cost and node numbers. As we can see from the figure, both protocols have the rising curves when nodes number increases. It is easy to understand when nodes number grows, more nodes needs to run the synchronization mechanism. But the transmission cost of proposed CTS protocol is much smaller than that of TPSN. Note that the overhead by TPSN increases sharply with the increase of the number of network nodes, while the increasing of CTS is more steady. This is because the bidirectional synchronization between a pair of nodes requires more complex operations and cost more energy. In CTS, when nodes number increases, as the hierarchy stays fixed, most nodes will be recruited as the cluster members which only receive the packets from corresponding cluster head, thus the energy cost for packet transmission will not increase much. Therefore we can conclude that the CTS is more suitable for large scale networks.

5 Conclusion

Time synchronization is important for most applications in wireless sensor networks. Existing protocols suffered from higher energy consumption or lower synchronization accuracy, especially for large scale networks. In this paper, we propose an energy efficient time synchronization protocol CTS, which is based on cluster mechanism. A cluster heads tree is constructed as the backbone first. Then the cluster head nodes and cluster members apply different time synchronization methods, so as to achieve high energy-efficiency. The performance of CTS is evaluated through theoretical analysis and simulation.

The proposed CTS protocol is designed for large scale wireless sensor networks with an idealized network environment. For future work, we are going to design synchronization protocols for applications in a more realistic environment, such as with signal interference or nodes mobility. Secondly, the hierarchy of the cluster heads tree is one of the most important factors for the efficiency of the protocol. So next we will also try to find out the best configuration of hierarchy and nodes number.

References

1. Zou, C., Lu, Y.: A time synchronization method for wireless sensor networks. In: Liu, B., Ma, M., Chang, J. (eds.) ICICA 2012. LNCS, vol. 7473, pp. 221–228. Springer, Heidelberg (2012)
2. Elson, J., Römer, K.: Wireless Sensor Networks: A New Regime for Time Synchronization. In: First Workshop on Hot Topics in Networks(HotNets-I), New Jersey (2002)
3. Elson, J., Girod, L., Estrin, D.: Fine-Grained Network Time Synchronization using Reference Broadcasts. In: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (2002)
4. Ganeriwal, S., Kumar, D., Srivastava, M.: Timing-sync protocol for sensor networks. In: Proc. of the 1st international Conference on Embedded Networked Sensor Systems, pp. 138–149 (2003)
5. Ping, S.: Delay Measurement Time Synchronization for Wireless Sensor Networks. IRB-TR-03-013 (2013)
6. Maróti, M., Kusy, B., Simon, G., Lédecz, Á.: The flooding time synchronization protocol. In: Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 39–49 (2004)
7. Greunen, J.V., Rabaey, J.: Lightweight time synchronization for sensor networks. In: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, pp. 11–19 (2003)

8. Nazemi Gelyan, S., Eghbali, A.N., Roustapoor, L., Yahyavi Firouz Abadi, S.A., Dehghan, M.: SLTP: Scalable Lightweight Time Synchronization Protocol for Wireless Sensor Network. In: Zhang, H., Olariu, S., Cao, J., Johnson, D.B. (eds.) MSN 2007. LNCS, vol. 4864, pp. 536–547. Springer, Heidelberg (2007)
9. Liu, X., Zhou, S.: Evaluation of several time synchronization protocols in WSN. In: International Conference of Information Science and Management Engineering (ISME), pp. 488–491 (2010)
10. Huang, A.Y., Zomaya, F.C., Delicato, P.P.: An accurate on-demand time synchronization protocol for wireless sensor networks. Journal of Parallel and Distributed Computing (2012)
11. Barnabas, K.L., Okeke, L.E.: Multi-level Clustering Architecture and Protocol Designs for Wireless Sensor Networks, Hawaii (2008)
12. Kong, L., Wang, Q., Zhao, Y.: Time synchronization algorithm based on cluster for WSN. In: The 2nd IEEE International Conference on Information Management and Engineering, pp.126–130 (2010)
13. Noh, K.L.: New advances in designing energy efficient time synchronization schemes for wireless sensor networks (2012)
14. Shang, Y.Y., Wan, Z., Zhang, X.H., Jing, Z.G., Xiong, X.: The Research of WSN Time Synchronization Algorithm Based on Clustering Network. In: Applied Mechanics and Materials, pp. 1340-1345 (2012)
15. Briff, L.R., Vega, A.L., Vargas, F.: On the Trade-off between Power Consumption and Time Synchronization Quality for Moving Targets under Large-Scale Fading Effects in Wireless Sensor Networks. Communications and Network 5, 498–503 (2013)
16. Akhlaq, M., Sheltami, T.: RTSP: An Accurate and Energy-Efficient Protocol for Clock Synchronization in WSNs. IEEE Transactions on Instrumentation and Measurement 62(3) (2013)
17. Bae, S.-K.: Performance evaluation of time synchronization protocols for wireless sensor networks. In: Park, J.J(J.H.), Jeong, Y.-S., Park, S.O., Chen, H.-C. (eds.) EMC Technology and Service. LNEE, vol. 181, pp. 651–658. Springer, Heidelberg (2012)

A Fast CABAC Algorithm for Transform Coefficients in HEVC

Nana Shan, Wei Zhou, and Zhemin Duan

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
helensnn@hotmail.com

Abstract. Context-based Adaptive Binary Arithmetic Coding (CABAC) is a method of entropy coding which is widely used in the next generation standard of video coding called High Efficient Video Coding (HEVC). It processes the amount of transform coefficients. Due to the complexity of CABAC, it accounts for the significant portion of the whole coding efficiency. Based on the feature of CABAC and transform coefficients, a fast CABAC algorithm is proposed in this paper by reducing the number of bins to be processed. Experiment results show that the proposed method achieves about 5.99% times saving in average of QP=17 with little performance degradation compared with the CABAC algorithm in test model HM-6.0.

Keywords: CABAC, transform coefficients, entropy coding, HEVC, bypass coding.

1 Introduction

Entropy coding is a kind of lossless coding method which is adopted in the last stages of video encoding. It converts the data of video to bits stream which are suitable to be transmitted and stored. Comparing with other methods of entropy coding, Context-based Adaptive Binary Arithmetic Coding (CABAC) can get high compression efficiency very close to the theoretical limit by taking full advantage of the feature of CABAC and the dependency of the data. It also maps the data to a certain decimals and updates the probability adaptively. So it is currently being the unique algorithm in HEVC entropy coding.

While CABAC provides high coding efficiency, it needs to process a series of syntax elements, and its coding speed is restricted by high computational complexity. The problem of how to improve throughputs of CABAC is becoming more and more emergency.

Several techniques are used to improve CABAC, it can be described as two ways: 1) improving the speed of coding by reducing the coding complexity [1][2][3], 2) improving the coding efficiency by adopting parallel processing [4][5][6].

Based on the analysis of the CABAC coding flow and its feature, a fast algorithm will be proposed in this paper. The remainder of this paper is organized as follows. Section 2 briefly introduces CABAC entropy coding. Section 3 provides a fast

CABAC algorithm. Section 4 presents the result of the proposed algorithm and compares it with the original algorithm. The conclusion is given in section 5.

2 Overview of Cabac

2.1 Process of CABAC

Fig. 1 shows the CABAC encoding process. It can be described as follow:

Binarization: In binarization stage, CABAC uses a bits stream with only "0" or "1" to encode the data of video by converting the non-binary value to the binary value. Several binarization forms are adopted such as unary, truncated unary, k-th order Exp-Golumb, and fixed length. The binarization form is selected based on the type of syntax element. And some combinations of them are often used. This process may be skipped when the initial value is binary. The output of the binarization is the mapped bins of syntax elements.

Context Modeling: Context model is a probability model for one or more bins of binarized symbol. The probability of context model is adaptive by the value of the previously coded bins. Bins with similar distributions often share the same context model. Once the encode process of one bin is over, the context model may switch from one to another.

Determination: There are two ways to process arithmetic coding, regular mode and bypass mode. It can be determined by the probability of bins.

Regular: CABAC set a range with initial value of 0 to 1, and divided it to two subintervals by the probability of the current encode bin. Then it selects one of the subintervals which the value of bin belongs to, and divided it to two subintervals by the probability of the next bin. After several divide and select, a certain range can be obtained. This process will be repeated until it gets a certain range. Finally, any decimal in this range can represent the input bins stream approximately.

Bypass: Bypass coding is a coding mode of CABAC for reducing its complexity. It is used to encode a symbol having equal probability by assuming probability of 0.5 for each bin value. When bypass mode is selected, the modeling process of the input bins is simply skipped. For bypass coding, the division of the range can be done by a shift, which will make it easier to be process than the regular recursion way.

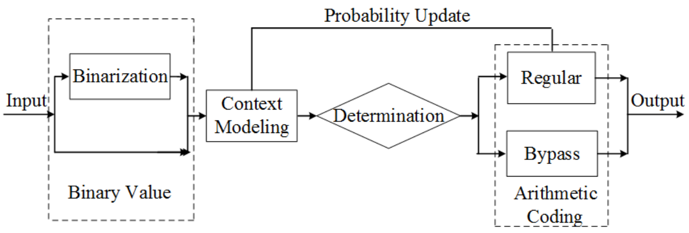


Fig. 1. Process of CABAC

2.2 The Feature of CABAC

2.2.1 Arithmetic Coding

The essence of arithmetic coding is it maps a decimal not for a single bit but for a whole input bits stream. And the principle of arithmetic coding is based on the recursive sub-division of interval selection. Once the original interval is set to [0, 1], and the sub-interval are repeated, according to the probability, the length of the range becomes more and more precise. In other words, the value becomes more and more close to its optimum. The division process can be described by diagram Fig. 2.

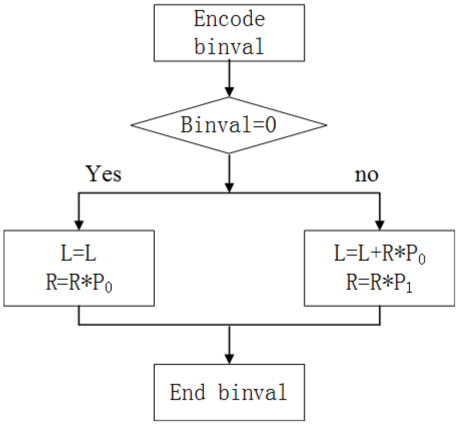


Fig. 2. Diagram of Arithmetic Coding

2.2.2 Self-Adaptive Update

The probability distribution of the input bits stream is variable. It changes with the dependency of the coded bits. Using binval as the current encoding bit and takes Z as the former bits stream have been encoded. Then the probability of current bit is the conditional probability that equals to P(binval|Z). CABAC trade-off the complexity and the coding efficiency by set a table of context. The update of probability is based on some principles [7].

3 The Proposed Fast CABAC Algorithm

3.1 The Bottleneck of CABAC

CABAC has essential bottleneck in the video coding. The coding efficiency of CABAC is determined based on the number of binary bins that it can process in a certain time. And the dependencies of the data make it very difficult to process in parallel.

3.2 The Proposed Algorithm

The video coding unit bases on the block. They can be described with coding unit (CU), predict unit (PU), and transform unit (TU). Rather than sending all the information about the video, CABAC only transmit the transform coefficients to reduce the number of the bits. And the processing of transform coefficients accounts for the significant portion of the whole coding efficiency.

However, in 4×4 TU, after series operations of prediction, transformation, and quantization, it can be represented in few non-zero transform coefficients. And those non-zero transform coefficients are concentrated on the top left corner. That means in a 4×4 TU, there are few coefficients to be transmitted.

The proposed algorithm of CABAC in this paper is mainly focus on how to remove the dependence of the data and reduce the transmitted bits in 4×4 TUs.

Fig. 3 shows an example of transform coefficients in a 4×4 TU. The scan order of these transform coefficients is based on a certain kind of scan mode. In this case, it adopts diagonal scan mode and the encoded symbols for these coefficients is in inverse scan order.

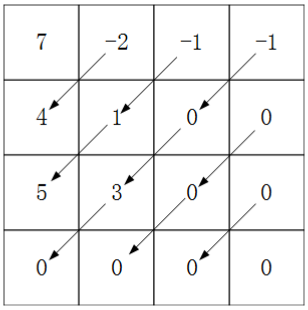


Fig. 3. Transform Coefficients in a 4×4 TU

In HEVC, the processing of these transform coefficients are described as following:

last_significant_coeff_flag_x (LSCFX) and last_significant_coeff_flag_y (LSCFY): At first, the algorithm transmits the value of coordinate for the last significant coefficient position in the TU.

significant_coeff_flag (SCF): If the current position is not the last significant coefficient, the algorithm transmits a significant_coeff_flag to indicate whether the transform coefficient is non-zero.

coeff_abs_level_greater1_flag (ONE): CABAC using ONE to indicate whether the absolute value of transform coefficient is greater than 1.

coeff_abs_level_greater2_flag (ABS): CABAC also using ABS to indicate whether the absolute value of transform coefficient is greater than 2 with ONE as 1.

coeff_abs_level_remaining (REM): When the transform coefficient is greater than 2, REM indicates the remaining value of the transform coefficient's absolute value minus 3.

coeff_sign_flag (SIGN): Coeff_sign_flag shows sign information of the non-zero transform coefficients.

Table 1. Transform coefficients in HEVC

No.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COEFF	0	0	0	0	0	0	-1	0	3	0	-1	1	5	-2	4	7
LASTX	3															
LASTY	0															
SCF	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	1
ONE	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1
ABS	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1
REM	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	4
SIGN	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0

Sending the X, Y position rather than LSCF can avoid the data dependency of LSCF and SCF. And it makes LSCF and SCF be processed in parallel.

Considering all the feature of transform coefficients, several technic can be taken to obtain further efficient improving. As mentioned earlier, there are many zero transform coefficients in the TUs. The SCF, ONE and ABS are encoded with adaptive context models, which will be the main bottlenecks of HEVC throughput. It is not necessary to use lots of bits to transmit all of these zero transform coefficients. To reduce the number of context coded bins, Jianle Chan suggested to process only 8 ONE and 1 ABS [8].

There only a few non-zero transform coefficients of ONE, even fewer of ABS in a 4×4 TU. As shown in tables 9-29 and 9-30 in JCTVC-J1003 [9], the initial process of ABS is much simpler than ONE. A fast CABAC algorithm for transform coefficients is proposed by skipping ABS to further reduce the number of coding bits. 16 context coded ABS in the worst case will be removed. And then, the values of REM are changed correspondingly to account for the missing ABS. The throughput can be improved by reducing the number of context coded bins and using bypass coded bins.

4 Experiment Results

4.1 The Parameter of Experiments

Two kinds of experiments were designed to verify coding efficiency of the proposed algorithm. In the first one, three sequences of different classes were selected by setting three configuration files with QP=32: encoder_randomaccess_main.cfg, encoder_lowdelay_main.cfg, encoder_intra_main.cfg. In the second one, three sequences of different classes were selected by setting one configuration files encoder_intra_main.cfg with 4 different QPs (17, 22, 27, 32) respectively. In order to compare coding performance of the proposed method and the original methods, we encode 100 frames for each condition. And the experiments adopted HM-6.0[10] as the test model.

4.2 The Simulations Results

Comparison between the proposed algorithm and original algorithm in test model HM-6.0 are shown in TABLE 2 and TABLE 3. According to the result in TABLE 2, the average time reduction of three sequences is 2.68%, 2.25%, and 5.41%.

The average time reduction of the configuration files (encoder_intra_main.cfg, encoder_lowdelay_main.cfg and encoder_randomaccess_main.cfg) is 1.48 %, 6.38%, and 2.49% respectively. The time reduction of proposed method is about 3.45% in average.

The time reduction of sequence crowd_run is much higher than others. And within three configuration files, using encoder_ lowdelay _main.cfg can obtain the best result.

Table 2. The comparisions of coding time with different configuration files

sequences	Times(s)	intra	lowdelay	randomaccess
foreman	original	218.06	665.58	470.15
	proposed	216.74	637.92	454.80
basketballdrill	original	3364.15	9437.70	6951.27
	proposed	3350.54	8914.79	6895.14
crowd_run	original	8707.32	27944.41	17694.61
	proposed	8409.55	25307.11	17096.47

Table 3. The comparisions of coding time with different Qps

sequences	Times(s)	QP=17	QP=22	QP=27	QP=32
foreman	original	302.43	265.82	237.08	218.06
	proposed	278.91	252.35	227.63	216.74
basketballdrill	original	5037.23	4481.28	3815.11	3364.15
	proposed	4698.69	4170.06	3623.44	3350.54
crowd_run	original	11896.76	10748.36	9764.41	8707.32
	proposed	11483.83	10339.61	9520.57	8409.55

The second experiment tested three sequence by setting one configuration files encoder_intra_main.cfg with 4 different QPs (17, 22, 27, 32) respectively. TABLE 3 shows the result of coding time reduction.

It shows the coding time reductions of three sequences are getting higher with the decrease of QP. And the average time reduction of the different QPs is 5.99%, 5.27%, 3.84% and 1.48%. When QP is 17, the result is the best.

Either adopting a higher quality video or using a lower QP, the number of the coding bits and the coding time of those processes will be increased. To sum up the result of these experiments, the time reductions of the proposed algorithm will be improved by the increase of the number of coding bits as shown in Fig. 4 and Fig. 5.

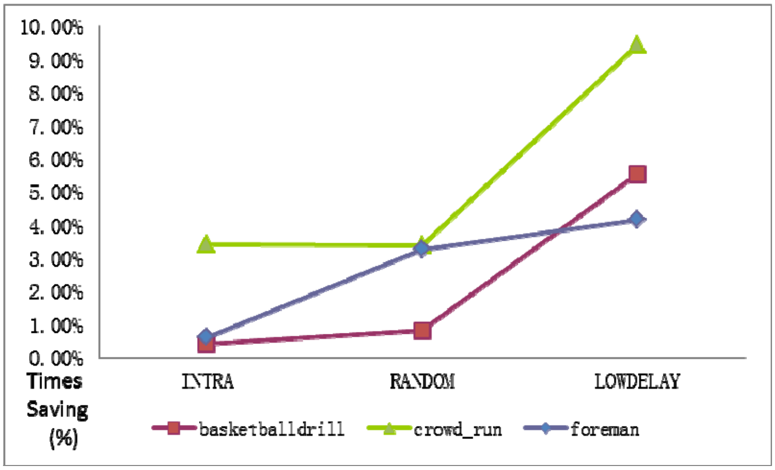


Fig. 4. Saving of Coding Time in Different Cfgs(%)

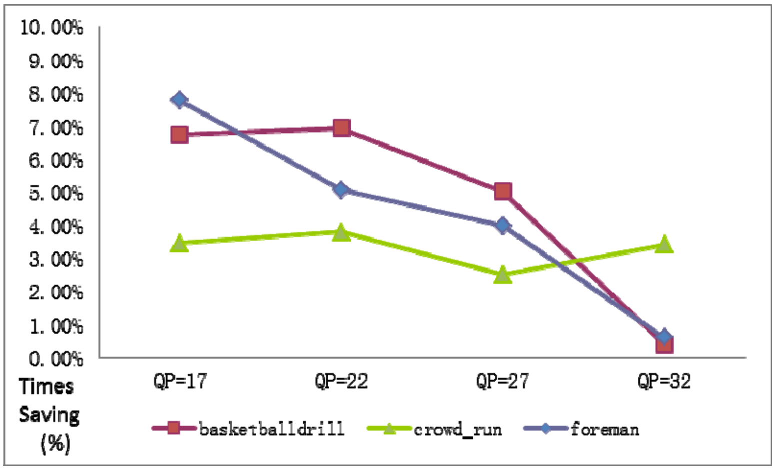


Fig. 5. Saving of Coding Time in Different QPs(%)

5 Conclusions

In this paper, a fast CABAC algorithm for transform coefficients in HEVC is proposed. The algorithm can greatly accelerates the speed of CABAC by reducing the number of bins to be processed. The proposed algorithm shows an almost negligible effect on the video quality and bit-rate. This makes it very suitable to process the huge data of high quality video in the future.

Acknowlegments. This work is supported in part by the National Natural Science Foundation of China (60902101) and New Century Excellent Talents in University (NCET-11-0824) of Ministry of Education of the People' Republic of China.

References

1. Hattori, R., Sugimoto, K., Itani, Y.: Fast bypass mode for CABAC. In: IEEE Picture Coding Symposium (PCS), pp. 417–420 (2012)
2. Nguyen, N., Ji, T., He, D.: Multi-level significant maps for large transform units. JCTVC-G644. Geneva (2011)
3. Ding, W., Xu, J., Che, X., Shi, Y., Yin, B.: Test Results on Context simplification for coefficients entropy coding. JCTVC-G301.Geneva (2011)
4. Sze, V., Chandrakasan, A.P.: A highly parallel and scalable CABAC decoder for next generation video coding. In: IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp. 126–128 (2011)
5. Coban, M., Chien, W.J., Karczewicz, M.: CU skip and split CABAC context modification for flexible wavefront parallel parsing. JCTVC-J0279. Stockholm (2012)
6. Chien, W.J., Sole, J., Guo, L., Karczewicz, M.: Context assignment for parallel coefficient level coding. JCTVC-J0304. Stockholm (2012)
7. Marpe, D., Schwarz, H., Wiegand, T.: Context-based adaptive binary arithmetic coding in the H. IEEE Trans. on CSVT 264/AVC Video Compression Standard. 13(7), 620–636 (2003)
8. Chen, J., Chien, W.J., Joshi, R., Sole, J., Karczewicz, M.: throughput improvement on CABAC coefficients level coding. JCTVC -H0554 (2012)
9. Bross, B., Han, W.J., Ohm, J.R., Sullivan, G.J., Wiegand, T.: High Efficiency Video Coding (HEVC) text specification draft 8. JCTVC-J1003 (2012)
10. HEVC Test Model, HM 6.0., https://hevc.hhi.fraunhofer.de/svn/svnHEVCSoftware/tags/HM-6.0

A Improved PageRank Algorithm Based on Page Link Weight

Xinsheng Wang, Jianchu Ma, Kaiyuan Bi, and Zhihuai Li

School of Information Science and Technology,
Dalian Maritime University,
Dalian, Liaoning, China
wangxinsheng@dlmu.edu.cn

Abstract. Page sorting has a great meaning to search engines. PageRank algorithm, based on random surfing model, has not fully taken the content of pages into consideration and the probability of links is supposed to be equal. Thus, this will lead to the ignorance of other important information from PageRank algorithm and its values of calculation are difficult to reach high accuracy. According to the analysis of the deficiencies mentioned above, a key property is supposed to be defined: randomization matrix and then proposing an improved algorithm called Pro-PageRank algorithm which does not simply weight the links equally but take the differences of weights into account. It can effectively solve the following problems: outlink pages with high PageRank values and users urgency for needed pages. According to Simulated experimental data, the improved algorithm, Pro-PageRank algorithm, compared with the traditional one, has more accuracy and further improves the quality of page rank so as to meet authority requirements of the page calculation results.

1 Introduction

With the advent of Web2.0 era[1], the development of the Internet is more intelligent, humane and socialized. It continues to affect and change people's ways of life and becomes the main source of information for people as well. Web2.0 emphasizes interactions between users, with particular emphasis on people-center. In Web2.0, users can fully demonstrate their personalization features. Meanwhile, a sharp increase in the amount of information on the network makes user demands for specific information become more and more urgent. So how to obtain information which is more relevant, authoritative and able to meet user demands in a higher accuracy rate in the vast and complex information environment has been the primary problem for scholars.

PageRank algorithm[2] is proposed by Google founders Larry Page and Sergey Brin in 1998 and later has been widely used in search engines. In view of the original PageRank algorithm existing 'topic drift' phenomenon[3], a number of domestic and foreign scholars have proposed some improved algorithms, such as the PageRank algorithm based on theme of sensitivity proposed by Taher

Haveliwa[4]. This algorithm calculates different themes for different values of PR, but the absence of context could lead to 'topic drift'. QD-PageRank algorithm[5] combines link relationships with web content but the time complexity and space complexity of the algorithm make it less practical. On the premise of Web situation changing at any time, in order to update webpage PageRank values quickly, Bahmani[6] and others further study the dynamic figure PageRank calculation. Gao Bin[7] and others, considering the basic meta-information and information on human oversight, propose a semi-supervised PageRank algorithm that can accurately calculate website link rankings.

This paper selects PageRank algorithm as a foundation for the algorithm of the study and proposes a suitable improved algorithm for the lack of PageRank algorithm-Pro-PageRank algorithm, which considers weights and relationships of links rather than simply weight equally. Comparing the two algorithms through experiments proves improved Pro-PageRank algorithm is more convincing and accurate than traditional PageRank algorithm.

2 PageRank Algorithm Analysis

PageRank algorithm is used by Google to identify the importance of a webpage, as well as key indicators to measure a website usability.Its core ideology[8] is: The importance of a page is decided by page link relations. When there is a link from page A to page B, it can be used as 'page A to page B casting a vote', thereby increasing the importance of page B.If viewing the network as a directed graph G = (V, E), where V represents the collection in the network web p_i and E represents the set of edges, there is a node p_i connecting to node p_j edge when linking from page to page.The iterative equation to calculate values of PageRank pages is:

$$PageRank(p_i) = (1 - d) + d \sum_{p_j \in M(p_i)} \frac{PageRank(p_j)}{C(p_j)} \tag{1}$$

equation: PageRank(p_i) is the representative of a webpage p_i's value PR. In general, PageRank(p_i) initial value takes 1; d is the damping coefficient which is a constant between 0 and 1, usually 0.85; M(p_i) is a collection of inward webpages (p_i); C(p_j) is the number of outward webpages p_j. By the equation can we know that to compute a page PR value is an iterative process and the accuracy of final calculation results is determined by the selection of initial values and the number of iterations.

PageRank algorithm is not based on websites to sort but on link structures between webpages to evaluate and rank the importance of a webpage which makes PageRank algorithm ignore other important information, causing it difficult to obtain a higher accuracy for sorting results. The theoretical basis of the algorithm is random surfing model[9]. This model's structure does not consider the content of webpages. It believes that the probability of a user outside link is random and equal without taking different pages as well as differences between dynamic needs and preferences of users into account. This will lead to an outdated page having a higher PageRank value. Meanwhile the urgent needs of the page the user want have not been well represented as well.

3 Improved Pro-PageRank Algorithm

The traditional PageRank algorithm does not consider every difference between link and link but those probabilities of outward page links are equal-namely, the average webpage PR value is assigned to its outward individual page links. However, in practical applications, links to other pages on the Web are not necessarily the same. They may be different, which require web link weights to be recalculated. The paper presents an improved PageRank algorithm called Pro-PageRank algorithm. Known as the name suggests, it is calculating PR values while considering the proportion of the number of Web links. The larger proportion indicates the more important this page is and the respective weights should be greater.

Definition 1(Link Relation). If webpage p_i links into webpage p_j, it is said that there exists link relationships between webpage p_i and webpage p_j, denoting link $(p_i,\ p_j)$. To mark link relationships between webpages, if there is a link that exists lines then let link $(p_i,\ p_j)$ = True; Conversely, link $(p_i,\ p_j)$ = False. Specifically, link (p_i,p_i) = False.

Definition 2(Page Relationship Matrix). The matrix consists of relations with all pages and the rest of each page, denoting L^T. Its equation can be expressed as

$$L^T = (link_{ij})_{n\times n} = \begin{pmatrix} link_{11} & \cdots & link_{1j} & \cdots & link_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ link_{i1} & \cdots & link_{ij} & \cdots & link_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ link_{n1} & \cdots & link_{nj} & \cdots & link_{nn} \end{pmatrix} \tag{2}$$

The Page relationship matrix records link relationships between all pages from a holistic perspective. In the equation, n represents the total number of pages. If there are link relationships between pages, the value of corresponding element is 1; Conversely, 0.

Definition 3(Page Outdegree). The number of line segments with a starting point-Page p_i, denoting $d^+(p_i)$.

Definition 4(Page Indegree). The number of line segments with a ending point-Page p_i, denoting $d^-(p_i)$.

Indegrees and outdegrees record partial information on a single page, as well as the two key attributes of this paper.Indegerees and outdegrees of pages will produce a comprehensive impact on web link weights. The traditional PageRank algorithm just does not consider this.

Definition 5(Page Indegree Matrix). The matrix formed by indegrees of all pages, denoting $\boldsymbol{Ind}$. Its equation can be expressed as

$$\boldsymbol{Ind} = (Ind_1, Ind_2, \cdots, Ind_i, \cdots, Ind_n) \tag{3}$$

Among them, $Ind_i = d^-(p_i)$.

Definition 6(1 Of Depth Page Indegree Matrix). Matrix consisting of the sum of indegrees of sub-pages of all pages,denoting $\boldsymbol{D}$. Its equation can be expressed as

$$\boldsymbol{D} = \boldsymbol{IndL} = (d_1, d_2, \cdots, d_i, \cdots, d_n) \tag{4}$$

Among them, $d_i = \sum_{j=1}^{n} ind_j \cdot link_{ij}$.

As can be seen from equation(3) and equation(4), 1 of depth Page indegree matrix is a product of the page indegree matrix and the page relationship matrix transpose. In fact, the paper considers how much chains of the page influence the page. The effect of proportion between pages will play a significant role. Due to the improved Pro-PageRank algorithm calculating PR values while considering the proportion of the number of Web links, the greater the proportion is, the larger corresponding weights should be. We define probability matrix to represent relationship proportions between webpages below.

Definition 7(Probability Matrix). Describe relationship proportions between webpages, denoting $\boldsymbol{Pro}$. Its equation can be expressed as

$$Pro = \begin{pmatrix} pro_{11} & \cdots & pro_{1j} & \cdots & pro_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ pro_{i1} & \cdots & pro_{ij} & \cdots & pro_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ pro_{n1} & \cdots & pro_{nj} & \cdots & pro_{nn} \end{pmatrix} \tag{5}$$

Among them, $pro_{ij} = \frac{link_{ji} \cdot ind_i}{d_j}$.

We use $\Omega_{in}(p_i)$ to represent collections of inlink pages, then $\Omega_{in}(p_i) = (p_{i1}, p_{i2}, \cdots, p_{ix})$, $x = d^-(p_i)$; $\Omega_{out}(p_i)$ represents collections of outlink pages, then $\Omega_{out}(p_i) = (p_{i1}, p_{i2}, , p_{iy})$, $y = d^+(p_i)$. When calculating PR values, after adding the proportion of the number of web links, the improved Pro-PageRank algorithm equation(1) becomes:

$$PageRank(P_i) = (1-d) + d \sum_{(P_j) \in \Omega_{in}(p_i)} \frac{PageRank(p_j) \cdot d^-(p_i)}{\sum_{(p_k) \in \Omega_{out}(p_j)} d^-(p_k)} \tag{6}$$

In practical applications, equations above are usually written in the form of matrix vectors. PageRank values are a matrix-vector, $R = \begin{pmatrix} PageRank(p_1) \\ \vdots \\ PageRank(p_n) \end{pmatrix}$

Thus, equation(6) can be written in matrix form as:

$$R = \begin{pmatrix} 1-d \\ \vdots \\ 1-d \\ \vdots \\ 1-d \end{pmatrix} + d \begin{pmatrix} pro_{11} & \cdots & pro_{1j} & \cdots & pro_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ pro_{i1} & \cdots & pro_{ij} & \cdots & pro_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ pro_{n1} & \cdots & pro_{nj} & \cdots & pro_{nn} \end{pmatrix} R \tag{7}$$

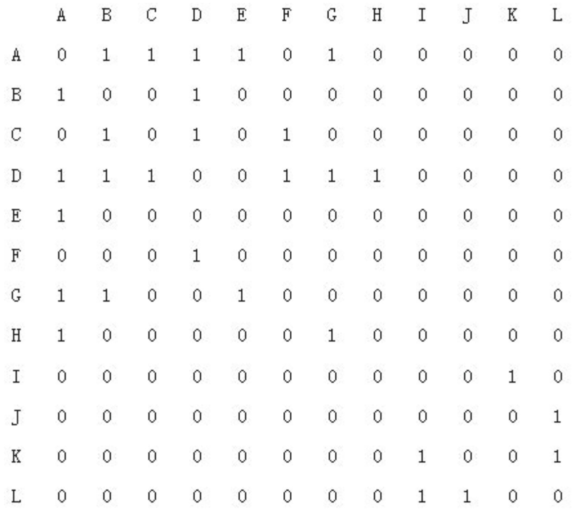
That is:

$$\boldsymbol{R} = \boldsymbol{C} + d \cdot \boldsymbol{Pro} \cdot \boldsymbol{R} \tag{8}$$

The calculation is an iterative process. First, assume an initial PR value; Then, evaluate each parameter in the equation until the results stabilize and get PR values.

4 Experiment Results and Analysis

The initial PR value is 1 for each page. Calculating K iterations by the improved Pro-PageRank algorithm can draw the final PR value of each page. Information of link structures shown in Figure 1 is assumed to analyze the improved Pro-PageRank algorithm that the chain matrix is randomly generated, which contains 12 pages.



	A	B	C	D	E	F	G	H	I	J	K	L
A	0	1	1	1	1	0	1	0	0	0	0	0
B	1	0	0	1	0	0	0	0	0	0	0	0
C	0	1	0	1	0	1	0	0	0	0	0	0
D	1	1	1	0	0	1	1	1	0	0	0	0
E	1	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	1	0	0	0	0	0	0	0	0
G	1	1	0	0	1	0	0	0	0	0	0	0
H	1	0	0	0	0	0	1	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	1	0
J	0	0	0	0	0	0	0	0	0	0	0	1
K	0	0	0	0	0	0	0	0	1	0	0	1
L	0	0	0	0	0	0	0	0	1	1	0	0

Fig. 1. chain matrix with 12 pages

Figure 2 and 3 show relations with the PR values and K iterations of traditional and improved algorithms. As can be seen from figures, when the time of iteration achieves 50, PR values using two algorithms tend to be a stable state.

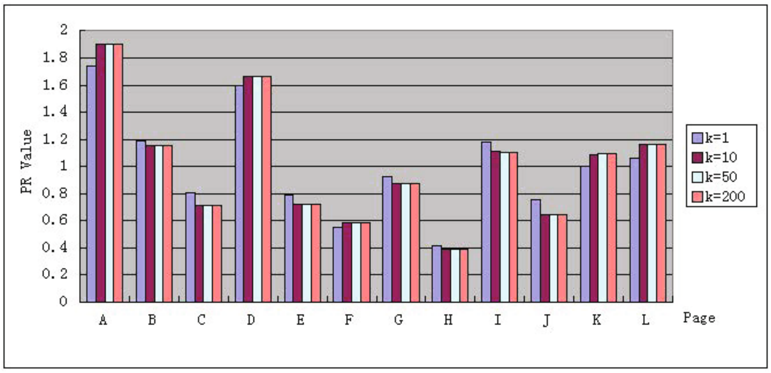


Fig. 2. chain matrix with 12 pages

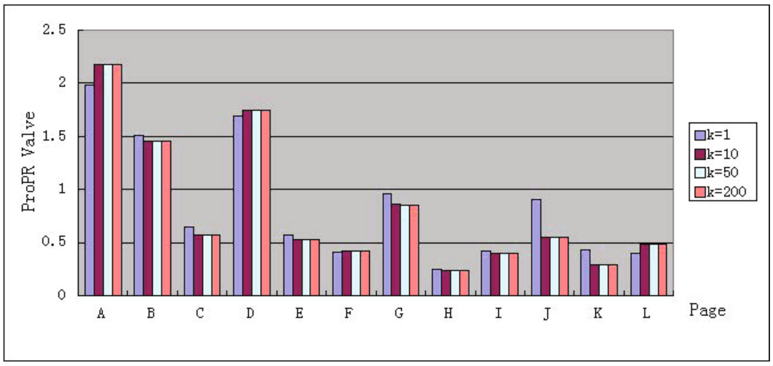


Fig. 3. chain matrix with 12 pages

Figure 4 shows the comparison of calculated PR values from the traditional and improved algorithm on condition that the time of iteration is under 50. Table 1 indicates relations of PageRank algorithm and the improved Pro-PageRank algorithm calculating PR values, webpage ranking and page indegrees.

It can be seen from Figure 4 and Table 1, page ranks calculated by two algorithms differ. Specifically, although page I and page L have two links at the same time , one of page I links is out of chains of page L so page I should be more popular than page L; There is a chain of page D pointing to page C but no chains point E and in accordance with 'If a page is referenced by important pages, the page is also important'[10], therefore page C is considered more important than page E, even though they have same number of links. Besides, it can be seen that Pro-PageRank algorithm is more reasonable and accurate than the traditional PageRank algorithm.

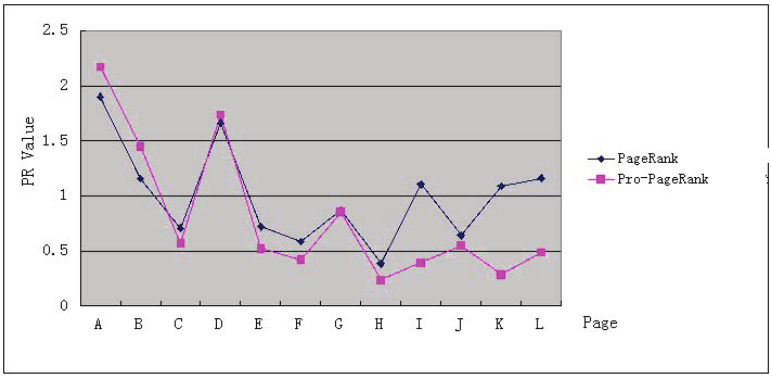


Fig. 4. chain matrix with 12 pages

Table 1. The relationship between indegrees and PR values of pages

PageRank	Indegree	PR Value	PR-Rank	ProPR Value	ProPR-Rank
A	5	1.902023276	1	2.172858036	1
D	4	1.665140168	2	1.741946987	1
B	4	1.157643734	4	1.449526733	1
G	3	0.873311357	7	0.856285160	1
C	2	0.709285992	9	0.570320714	1
J	1	0.643008455	10	0.548574628	1
E	2	0.720757430	8	0.524823578	1
I	2	1.106471163	5	0.486825186	1
L	2	1.160019894	3	0.486301416	1
F	2	0.586886304	11	0.421079550	1
K	1	1.090500488	6	0.288054930	1
H	1	0.385942036	12	0.237062513	1

5 Conclusions

The paper analyzes PageRank algorithm and its shortcomings, finding that PageRank algorithm relies on link structures between pages to evaluate and rank the importance of a webpage, meanwhile the random surfing model of the algorithm considers the probability of a user outside links is equal. But it does not take differences between different pages, user's needs and dynamic demands into account. In view of this, an improved Pro-PageRank algorithm is proposed based on the original PageRank algorithm. The improved PageRank algorithm considers weights of the relationship between links and the proportion of the number of Web links. The greater the proportion is, the more important the page is. And respective weights should be greater accordingly. Simulated experimental data show that PR values calculated by the improved Pro-PageRank algorithm are more accurate so that high popularity of pages can be better reflected. It can be seen that Pro-PageRank algorithm is more accurate than the traditional PageRank algorithm.

References

1. Wang, W.-J., Sun, J.: The summarization of research and application of web2. 0. Information Science 12, 30 (2007)
2. PageRank. PhD thesis
3. Hatakenaka, S., Miura, T.: Query and topic sensitive pagerank for general documents. In: 2012 14th IEEE International Symposium on Web Systems Evolution (WSE), pp. 97–101. IEEE (2012)
4. Haveliwala, T.H.: Topic-sensitive pagerank. In: Proceedings of the 11th International Conference on World Wide Web, pp. 517–526. ACM (2002)
5. Xing, W., Ghorbani, A.: Weighted pagerank algorithm. In: Proceedings of the Second Annual Conference on Communication Networks and Services Research, pp. 305–314. IEEE (2004)
6. Bahmani, B., Kumar, R., Mahdian, M., Upfal, E.: Pagerank on an evolving graph. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 24–32. ACM (2012)
7. Gao, B., Liu, T.-Y., Wei, W., Wang, T., Li, H.: Semi-supervised ranking on very large graphs with rich metadata. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 96–104. ACM (2011)
8. Jin-tao, J.I.A.O.: Improved web mining algorithm based on pagerank. Computer Engineering 15, 102 (2009)
9. Keong, B.V., Anthony, P.: Pagerank: A modified random surfer model. In: 2011 7th International Conference on Information Technology in Asia (CITA 2011), pp. 1–6 (2011)
10. Haveliwala, T.: Efficient computation of pagerank (1999)

Computation Offloading Management for Vehicular Ad Hoc Cloud

Bo Li, Yijian Pei, Hao Wu*, Zhi Liu, and Haixia Liu

School of Information Science and Engineering,
Yunnan University, Kunming, China

Abstract. Vehicular Ad Hoc Cloud(VACloud) research tries to combine vehicular ad hoc networks and mobile cloud computing, i.e., to implement cloud computing among vehicles interconnecting with wireless ad hoc networks. This paper proposed a computation offloading framework for VACloud, whose functional components including resource registration and discovery, application partition, surrogate selection, offloading monitoring,etc. One application partition method and four surrogate selection strategies were put forward and their performances were investigated via simulation experiments. It's shown that the multi-attributed strategy that considers multiple attributes, including the computation capacity and the distance, outperformed than the others, with higher task completion rate and shorter completion time.

Keywords: Vehicular ad hoc cloud, computation offloading, resource management, task partitioning, resource selection.

1 Introduction

Mobile cloud computing is a concept which combines cloud computing to the mobile environments. It is a product of the development of cloud computing and mobile communications[1][8][2]. The emergence of mobile cloud computing extends the application areas of cloud computing and offers a new way to augment the ability of resource-constraint mobile devices by connecting the abundant resources of cloud computing and mobile devices[1]. With the help of cloud resources(no matter they are remote resources in long-distance power data centers or local resource such as personal computers or notebooks, or even other smart phones nearby), the shortage of both computation capability and battery capacity of mobile devices can be overcome or relieved[1][7].

For the last few years, the Vehicular Ad-Hoc Network (VANET) has received much attention in both the product and the academic areas. A VANET is a set of moving vehicles that are connected via wireless networks and communicate with each other to enable novel and attractive solutions in areas such as vehicle and road safety. Vehicular cloud computing, as a new technical shifting, are trying to offer the advantages of cloud computing to the vehicles or the vehicle

* Corresponding author.

drivers of the VANET[5][6][9][12][3][10][11]. The objectives of vehicular cloud computing include to provide computational service at a low cost, to minimize traffic congestion, accidents, travel time and environmental pollution, and so on. Vehicular cloud computing provides a technically feasible incorporation of wireless communication technologies, intelligent traffic systems and mobile cloud computing for better and safer traffic systems.

Computation offloading is an important method to enhance the capability of resource-constrained devices by offloading a partial or whole application from the devices to other more powerful devices to enhance the computational ability or to lengthen battery capacity. Offloading solutions such as Cuckoo, MAUI, COMET, and ThinkAir offload applications via Wi-Fi or 3G networks to servers or commercial clouds such as Amazon EC2. However, Offloading applications in a mobile cloud computing environment is complicated by unreliable wireless links, node mobility, battery capacity and varying QoS requirements for applications. In such an environment, how to allocate offloading applications to available surrogates becomes an important issue.

On the basis of existing researches on mobile cloud computing, vehicular ad hoc networks and computation offloading, this paper investigated the problem about how to implement computation offloading in vehicular cloud computing environments, proposed a framework to support computation offloading. Further more, assume the vehicles are distributed and moved in a two-dimensional plane, this paper quantified the longest time between directly inter-connected vehicles and the maximum amount of computation that can be offloaded. On the basis, four surrogate selection strategies were proposed and their performances were investigated via simulation.

2 Computation Offloading in Vehicular Ad Hoc Clouds

These are two kinds of computation offloading in VACloud: Offloading between vehicles and offloading between vehicles and roadside units(RSU). Fig. 1 shows that vehicles A, B, C, D, E form a VACloud together. The tasks in one of the cars can be offloaded to other vehicles with available resources to perform. Vehicle nodes switch data with each other by direct or multi-hop communication links. If the vehicles are connected via wireless ad hoc networks, they can communicate without the help of roadside units. However, with the help of roadside units, the vehicles can not only offload applications to other vehicles nearby, but also to remote cloud resources. For example, vehicle D can offload its applications to vehicles C and E, which are connected with D via wireless ad hoc networks. On the other hand, vehicle D can also offload its applications to remote cloud resources in the Internet by using RSU AP2. Similar to VANET, here the former type of offloading is defined as *V2V offloading* and the latter *V2I offloading*. In some cases, if both vehicles C and D and remote cloud resources are available at the same time, vehicle D can even offload its applications to both kinds of resources simultaneously and jointly. In this paper,issues on V2V offloading are mainly considered and those of V2I offloading will be investigated in future researches.

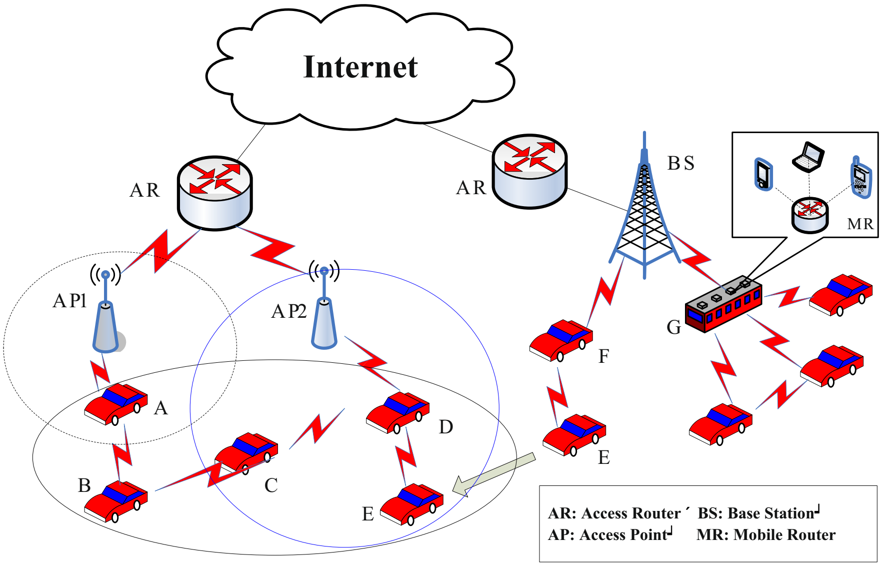

Internet
AR
AR
BS
MR
AP1
AP2
A
B
C
D
E
F
E
G
AR: Access Router BS: Base Station
AP: Access Point MR: Mobile Router

Fig. 1. Computation offloading in vehicular clouds

Based on the computation offloading scenarios in Fig. 1, a computation offloading framework in such computing environments is presented in Fig. 2. When a car needs to offload some application, the resource discovery module within the offloading manager of the car detects available surrogates nearby by sending out querying packets. Once some other cars nearby receive this request and will to provide offloading service, they will send back a response packet by the resource registration module, which including such information as the relative position of the vehicle, velocity, moving direction, computing power, etc. On the basis, the offloading decision module will select appropriate surrogate vehicles according to certain selection strategy and partition the application into local part and offloaded part according to certain partitioning methods. Finally, the local part will be executed on the car itself and the offloaded part, as well as data related, will be transferred to the selected surrogate cars and be executed. During the execution of the task, the runtime monitoring module in the surrogate vehicles will monitor the runtime status of the offloaded part continuously and report to the client vehicle if necessary.

2.1 Resource Discovery

This is one of key components and the basis of the overall offloading framework. Some researches have already been carried out to investigate the problem about how to discover resources in mobile cloud computing environments[2]. I n this paper, in order to increase the possibility of finding available resources, we assume the query packet is disseminated among vehicles in a flooding way, i.e.,

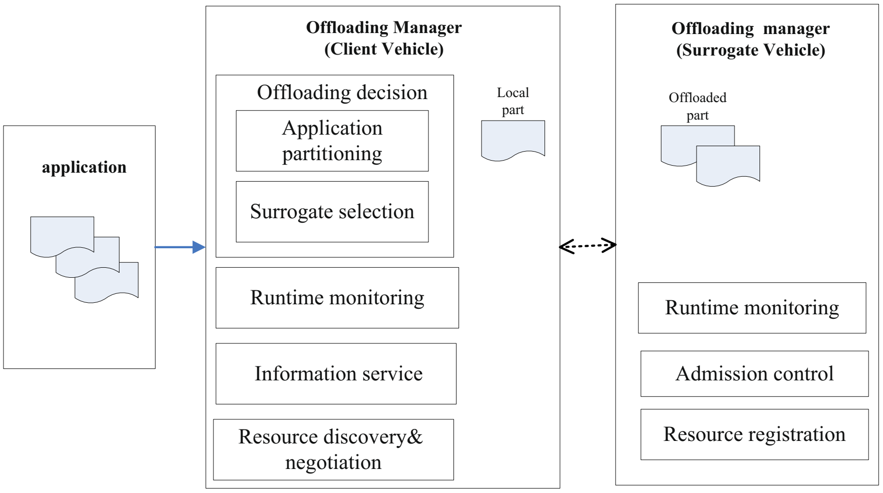


Fig. 2. Computational offloading framework in vehicle networks

the client vehicle sends out the same query packet to every vehicles nearby and once the packet is received by some other vehicle, it will be spreaded out again. Finally all vehicles receiving such a query packet will respond to the client vehicle with information such as the relative position of the vehicle, velocity, moving direction, computing power, etc. On the basis, the client vehicle can get following parameters for offloading decision.

C_i :Denotes the subtasks that the vehicle can perform per second.

V_i : Denotes the speed of the vehicle.

D : Denotes the direction of the vehicle, 0 represents the bottom-up, 1 represents a top-down, 2 represents right to left, 3 represents left to right.

R : Denotes signal coverage radius of a vehicle.

u : Denotes the hop counts from the client vehicle node to the target node.

P_{ij} : Denotes the communication overheads per task.

Q_i : Denotes the computing overheads for execution per task.

T_{ij} : Denotes the longest time between vehicles for communication. Including the communication time of a task from the client to the other vehicles($T_{ij}^{'}$) and the execution time of a task on other vehicles($T_{ij}^{''}$).

W_i : Denotes the amount of tasks that a vehicle can perform. That is the number of subtasks assigned to each vehicle. $W_i = C_i \times T_{ij}^{''}$,$W_0$ represents the task load that a client vehicle can complete. W_i represents the task load that other vehicles can complete. So the total size of the task $W = W_0 + \sum_{i=1}^{n} W_i$, $n = 1, 2, 3, ...$

The execution time of a task on other vehicles($T_{ij}^{''}$) can be calculated according to the vehicle speed and the distance between the vehicle client and the vehicles that can provide resources available. Then the amount of tasks that a vehicle can perform can be represented by the product of the computing power and communication time.

In the following, the procedure to calculate the longest time between directly inter-accessible vehicle pairs(i.e., T_{ij}) and the amount of tasks that a vehicle can perform(i.e.,W_i) are presented in detail.

2.2 The Longest Time between Inter-Accessible Vehicle Pairs

Assume the vehicles are moving in a two-dimensional plane(See Fig. 3). At the present time, assume the client vehicle node stays at the origin of coordinate, i.e., $(x_0, y_0) = (0, 0)$ and the coordinate of the surrogate vehicle node is (x_i, y_i). After time T_{0i}, the coordinates of the client node and the surrogate node become $(x_0^{'}, y_0^{'})$ and $(x_i^{'}, y_i^{'})$ respectively. When the communication distance between the two vehicle nodes reaches the coverage of the wireless communication technology, they will no longer directly inter-accessible in the next moment. The elapsed time during this process is defined as the longest communication time T_{ij}. The longest communication distance D_{ij} is the wireless coverage radius R:

$$R = \sqrt{(x_i^{'} - x_0^{'})^2 + (y_i^{'} - y_0^{'})^2} \tag{1}$$

Assume the vehicles just move horizontally or vertically in the two-dimensional plane and the relative movement direction can be divided into two cases: (1)move in the same horizontal/vertical direction; and (2) move at right angles.

Move in the Same Direction. At the present time, assume the client vehicle and the surrogate vehicle are on the same horizontal line and their coordinates are $(x_0, y_0) = (0, 0)$ and $(x_i, y_i) = (x_i, 0)$ respectively. After time T_{0i}, the coordinate of the client node becomes $(x_0^{'}, y_0^{'}) = (V_0 \times T_{i0}, 0)$, and the coordinates of the surrogate becomes $(x_i^{'}, y_i^{'} = (x_i + V_i \times T_{0i}, 0))$. Bring the values of the two coordinates into Equation 1, the value of T_{0i} can be obtained as follows:

$$T_{0i} = \frac{-x_i \pm R}{V_i - V_0} \tag{2}$$

As time is unlikely to be negative, thus the positive value is regarded as the longest time between the two vehicles to communication directly.

If the vehicles move in the same vertical direction, T_{0i} can be calculated in the same way.

Move at Right Angles. At the present time, assume the coordinates of the client vehicle and the surrogate vehicle are $(x_0, y_0) = (0, 0)$ and $(x_i, y_i) = (x_i, 0)$ respectively. If the client vehicle and the surrogate vehicle move at right angles, after time T_{0i}, the coordinate of the client node becomes $(x_0^{'}, y_0^{'}) = (V_0 \times T_{0i}, 0)$, and the coordinate of the surrogate vehicle becomes $(x_i^{'}, y_i^{'}) = (x_i, V_i \times T_{0i})$. Bring the values of the two coordinates into Equation 1, the value of T_{0i} can be obtained:

$$T_{0i} = \frac{-(y_i \times V_i - x_i \times V_0) \pm \sqrt{(y_i \times V_i - x_i \times V_0)^2 - \left({V_i}^2 + {V_0}^2\right)\left({x_i}^2 + y_i^2 - R^2\right)}}{{V_i}^2 + {V_0}^2}$$

Take client vehicle A and surrogate vehicle E for example, as shown in Figure 3, the coordinate of vehicle E is (x_4, y_4). The point O is the center of the crossroads, and the longest communication time between vehicle A and E is T_{04}. After time T_{04}, vehicle A moves horizontally to $A^{'}(x_0^{'}, 0)$ from left to right and vehicle E moves to $E^{'}(x_4^{'}, y_4^{'})$ from bottom to top. While they are moving, if the distance between them is within their coverage, they will be able to communicate with each other. However, once the distance is beyond the coverage, they can not communicate directly.

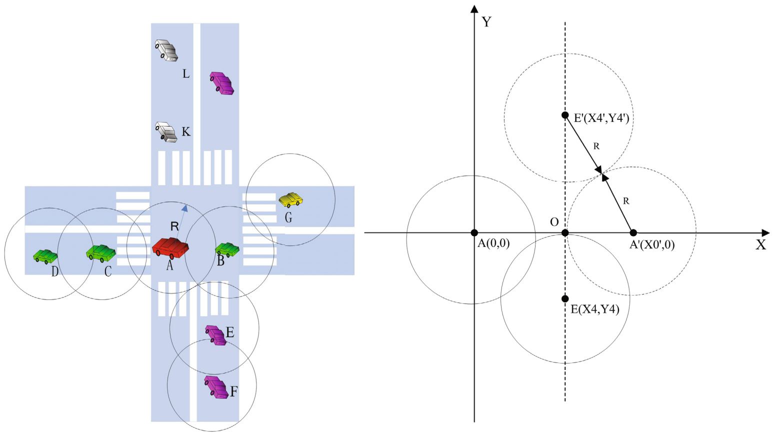


Fig. 3. The motion model of two vehicles that has a 90-degree angle

Furthermore, assume $(0, 0)$ and (x_4, y_4) and $(x_0^{'}, 0)$ and $(x_4^{'}, y_4^{'})$ are the beginning and ending coordinates of A and E respectively at which they can begin to communicate and cannot communicate anymore, the longest communication time between vehicle A and E can be figured out as follows:

$$T_{0i} = \frac{-(y_4 \times V_4 - x_4 \times V_0) \pm \sqrt{(y_4 \times V_4 - x_4 \times V_0)^2 - \left({V_4}^2 + {V_0}^2\right)\left({x_4}^2 + y_4^2 - R^2\right)}}{{V_4}^2 + {V_0}^2}$$

2.3 The Amount of Tasks That a Vehicle Can Perform

The longest communication time between the client vehicle node and the surrogate vehicle nodes (i.e., T_{0i}) includes the communication time of the task between them (i.e., $T_{ij}^{'}$) and the execution time of the task on the surrogate vehicle($T_{ij}^{''}$), ie $T_{0i} = T_{0i}^{'} + T_{0i}^{''}$, so $T_{0i}^{''} = T_{0i} - T_{0i}^{'}$.

The computing power of the surrogate vehicle is known as C_i, so the maximum amount of tasks that a vehicle can perform(W_i) can be calculated as follows: $W_i = C_i \times T_{0i}^{''}$.

3 Surrogate Vehicle Selection

On the basis of the information aforementioned, if there is a task that need to be offloaded from the client vehicle to surrogate vehicles, it's necessary for the schedule to decide how to select surrogate vehicles to execute the offloaded task. In this paper, four decision strategies were put forward and their performances were evaluated in the following.

– Random selection strategy: During surrogate selection, random selection strategy first selects a vehicle node randomly, then calculates the amount of task that this node can perform. If this node is able to perform the task completely, then the task will be assigned to it. If this node cannot perform the task completely, then a node will be selected randomly in the remaining nodes and the amount of task that the new node can perform will be worked out. Repeat this process until one node is selected to offload the task.
– Computing capacity-based selection strategy: Sort available surrogate vehicle nodes in descending order according to their computing capacity C_i, then select a vehicle node in this order until one node is found out to perform the task completely.
– Distance-based selection strategy: Sort the vehicle nodes according to their distance from the client node in ascending order and assign the task to the vehicle nodes in this order.
– Multi-attributed selection strategy: Multi-attribute algorithm determines the final node to perform the tasks by considering multiple parameters of the surrogate vehicle nodes. This algorithm can select the best node from the candidate surrogate nodes by considering the variety of factors involved in the offloading process.
  During surrogate selection, the factors need to be considered are as follows: the computing capacity of vehicles, the longest communication time between vehicles, the communication overheads for transmission per task, the computing overheads for execution per task. All of these factors constitute a row vector: $\boldsymbol{a} = [C_i, T_{ij}, P_{ij}, Q_i]$. After getting the parameter matrix consisting of the parameter vector for each surrogate vehicle node, filtrate out the vehicles and sort them in descending order.

4 Simulation and Results

The simulation environment was built on the Eclipse development platform in Java language. It is set up by referring to the classic development mode of the

MVC three layer architecture. The simulation environment is mainly composed of three modules: the Car, the Action, and the ControlLayer.

The Car module is a model of vehicles, which has all the attributes required for vehicles in the experiment. These attributes include the definition of car properties and the constructor of the vehicles. And it also provides: (1) Container function(For carrying the cars searched and processed by strategy) (2) Deduplication function(Avoid the searched car being searched again) (3) The display function of vehicle attributes and other related information.

The Control Layer is the core function layer of this architecture, its main functions are as follows: (1) To produce a certain number(this number is variable) of vehicles by their basic attributes and put these cars into the Car Container(Can be seen as the vehicle information database). This can imitate and search the information of vehicles located within the signal coverage of the client vehicle. (2) To provide the result set of four target selection strategies according to the computing power, the communication time, the relative distance, the communication overheads, the computational overheads and other related processing operations. In the simulation experiments, the calculation of the longest communication time and the selection of target vehicle nodes are realized in the Control Layer.

The Action module is the executive layer which can complete the execution of the function method of the Control Layer.

The performance metrics of the simulation experiments are divided into two issues: the total task completion time and the completion rates. These performance metric results of the four target selection strategies were obtained by changing the speed of the vehicle and the number of the target nodes. When the selected target node has completed all the tasks, these four target selection strategies will be evaluated according to the task completion time and task partition number of the total tasks. When the selected target node has not completed all the tasks, the performance of the four target selection strategies will be evaluated by the task completion rate.

In the first part, when the speed of the vehicles is changed, the task completion time, the task partition number and the task execution cost will change with it. Following are the simulation results of these three cases:

As can be seen from Figure 4, the task completion time decreases as the vehicle speed increases. This is because the longest communication time between the client vehicle and the target vehicles decreases when the vehicle speed increases. As a result, the tasks assigned to the target nodes are reduced. Thus the task completion time will be gradually reduced. Meanwhile, as is shown in figure 5, the task requires more target nodes to perform, thus making the task partition number increases with the increase of the vehicle speed. As the task partition number increases, the total cost of task execution increases accordingly. Thus as the speed of the client vehicle increases, the total cost of task execution is certain to go up(As is shown in figure 6).

Since the task allocation is based on the product of the computing power and the expected execution time on the target node, the task completion time of the selection strategy which is based on the computing capacity of vehicles is shorter(Figure 4) in the four target selection strategies. When the speed of the client vehicle reaches a certain size, the task completion time of random selection strategy and selection strategy based on the distance between vehicles shows a rapid decline(Figure 4) due to the rapid increase in task partition number(Figure 6). Since the task partition number of the multi-attribute selection strategy is the least(Figure 5), the task completion time is relatively longer(Figure 4). Meanwhile, because the task partition number is the least, so the desired number of the target nodes is also the least. This makes the cost of the task execution reduce to the minimum in the four target selection strategies(Figure 6).

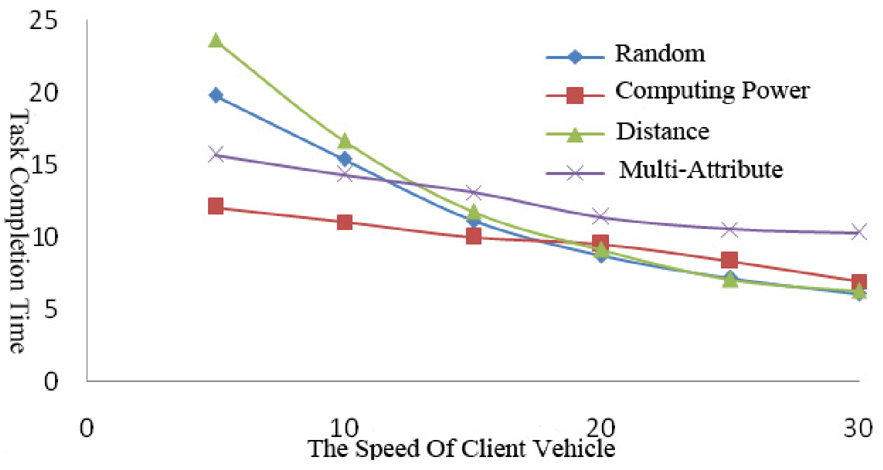


Fig. 4. The impact of speed changes on the task completion time

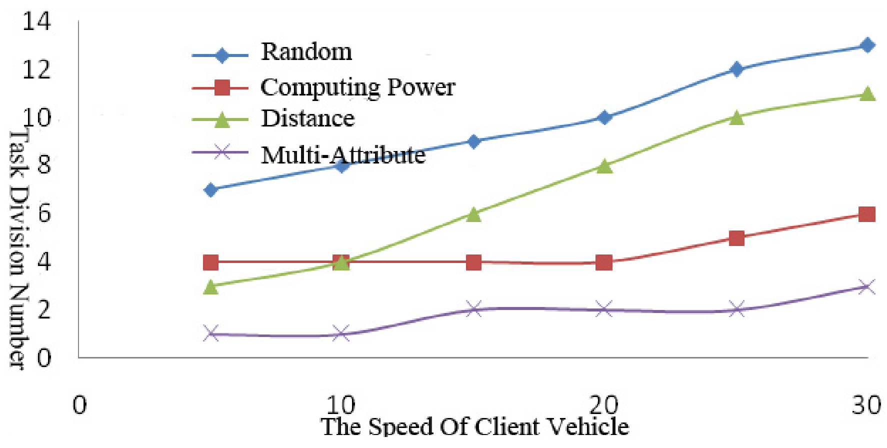


Fig. 5. The impact of speed changes on the task partition number

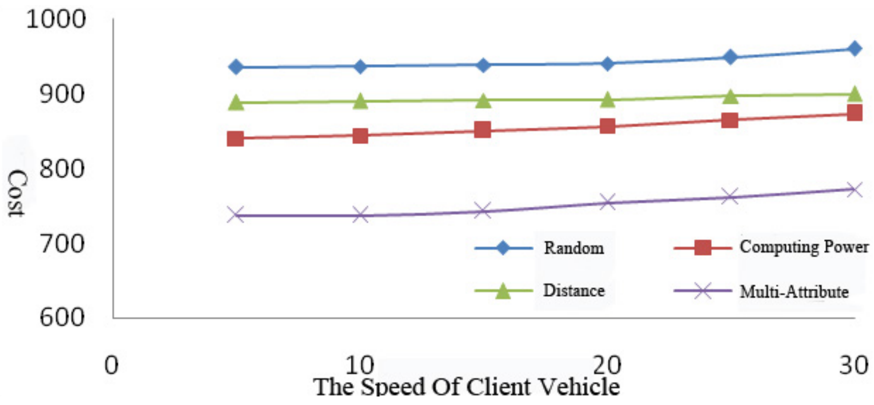


Fig. 6. The impact of speed changes on the task execution costs

As can be seen from the simulation results, the multi-attribute selection strategy has the best performance on both task partition number and task execution cost. However, the multi-attribute selection strategy needs more time to complete the task, while the selection strategy based on the computing capacity has the best performance on the task completion time.

In the second part, the moving speed of the client vehicle remains unchanged. The performance of the four target selection strategies varies as the target node number changes.

When the selected target node is not able to complete all the tasks, the performance of these four target selection strategies can be observed by task completion rate. When the target node number reaches to a certain value, the tasks can be completed. Then the performance of the four target selection strategies can be observed by the task completion time. This can be seen from figure 7 below.

As is shown in figure 7, with the increase of the target node number, the task completion rate of the four target selection strategies increases correspondingly. Under the condition of the same number of target nodes, the multi-attribute selection strategy has a higher task completion rate than the other three target selection strategies. Moreover, the multi-attribute selection strategy can complete all the tasks when the target node number is just four. However, the other three target node selection strategies need more to complete all the tasks.

5 Conclusion

Vehicular cloud computing, as an combination of vehicular ad hod network and cloud computing, is getting more and more attention from the industry and the academic area. The computational offloading technology of the vehicular ad hoc cloud can exploit idle computing resources and divide the task into several parts, then offload some of them to other resource nodes to perform. It makes full use of network resources scattered and enhance the computation ability of the vehicles.

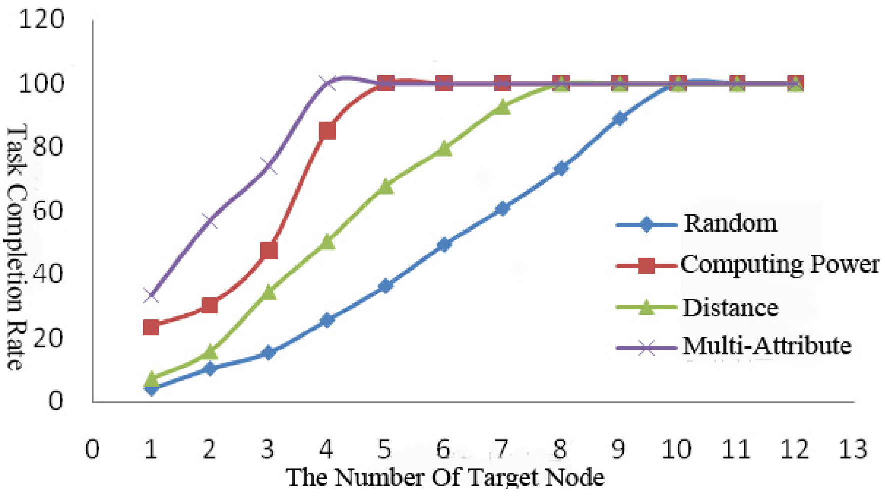


Fig. 7. The impact of target nodes changes on task completion rate

This paper proposed an architecture to implement computation offloading among vehicles inter-connected with ad hoc networks, defined the parameters needed and illustrated how to exchange these information between vehicles via query-respond packets. Then, assume the vehicles are moving in a two-dimensional plane, the longest time interval between inter-accessible vehicle pairs and the computation amount of each available surrogate vehicle are quantified. Based on such information, four surrogate vehicle selection strategies were proposed and their performances were evaluated via simulation. It's proved that the selection strategy that considers multiple attributes can outperform the others with respect to the completion time and the completion rate of the tasks.

In this paper, we only considers the computation offloading between directly inter-connected vehicles. In the future researches, multi-hop connection will be included. Further more, more realistic mobility scenarios, such as the Manhattan mobility model, will be considered.

References

1. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., Buyya, R.: Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. IEEE Communications Surveys & Tutorials 16(1), 337–368 (2014)
2. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing 13(18), 1587–1611 (2013)
3. Gerla, M., Weng, J.T., Pau, G.: Pics-on-wheels: Photo surveillance in the vehicular cloud. In: 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 1123–1127 (2013)
4. Gu, L., Zeng, D., Guo, S.: Vehicular cloud computing: A survey. In: 2013 IEEE Globecom Workshops (GC Wkshps), pp. 403–407 (2013)

5. He, W., Yan, G., Xu, L.D.: Developing vehicular data cloud services in the iot environment. IEEE Trans. Industrial Informatics 10(2), 1587–1595 (2014)
6. Hussain, R., Oh, H.: Cooperation-aware vanet clouds: Providing secure cloud services to vehicular ad hoc networks. JIPS 10(1), 103–118 (2014)
7. Kaewpuang, R., Niyato, D., Wang, P., Hossain, E.: A framework for cooperative resource management in mobile cloud computing 31(12), 2685–2700 (2013)
8. Khan, A., Othman, M., Madani, S., Khan, S.: A survey of mobile cloud computing application models. IEEE Communications Surveys & Tutorials 16(1), 393–413 (2014)
9. Lee, E., Lee, E.K., Gerla, M., Oh, S.Y.: Vehicular cloud networking: architecture and design principles. IEEE Communications Magazine 52(2), 148–155 (2014)
10. Mershad, K., Artail, H.: Crown: Discovering and consuming services in vehicular clouds. In: 2013 Third International Conference on Communications and Information Technology (ICCIT), pp. 98–102 (2013)
11. Mershad, K.W., Artail, H.: Finding a star in a vehicular cloud. IEEE Intell. Transport. Syst. Mag. 5(2), 55–68 (2013)
12. Whaiduzzaman, M., Sookhak, M., Gani, A., Buyya, R.: A survey on vehicular cloud computing. J. Network and Computer Applications 40, 325–344 (2014)

An Approach to Model Complex Big Data Driven Cyber Physical Systems

Lichen Zhang[1,2]

[1] Faculty of Computer Science and Technology
Guangdong University of Technology
Guangzhou 510090, China
[2] Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
Shanghai 200062, China
zhanglichen1962@163.com

Abstract. Big data driven cyber physical systems not only meet big data 4V feature requirements, but also have to meet time constrains and spatial constraints of cyber physical systems. Big data driven cyber physical systems have to deal with time-constrained data and time-constrained transactions. They are now being used for several applications such as automobile and intelligent transportation systems, aerospace systems, medical devices and health care systems in each of big data driven cyber physical applications, data about the target environment must be continuously collected from the physical world and processed in a timely manner to generate real-time responses. Those systems contain a large network of sensors distributed across different components, which leads to a tremendous amount of measurement data available to system operators. Regarding big data modeling, an important question is how to represent a moving object. In contrast to static objects, moving objects are difficult to represent and model. The efficiency of modeling methods for moving objects is highly affected by the chosen method to represent and analyze the continuous nature of the moving object. The design of big data driven cyber physical systems requires the introduction of new concepts to model classical data structures, 4V features, time constraints and spatial constraints, and the dynamic continuous behavior of the physical world. In this paper, we propose a model based approach to model big data driven cyber physical systems based on integration of Modelica, Modelicaml, AADL, RCC and clock theory, we illustrate our approach by specifying and modeling Vehicular Ad hoc Networks (VANET).

Keywords: Big data, CPS, Modelicaml, RCC, VANET.

1 Introduction

Big Data is characterized by what is often referred to as a multi-V model, variety, velocity, and volume are the items most commonly mentioned. Variety represents the data types, velocity refers to the rate at which the data is produced and processed, and volume defines the amount of data. Veracity refers to how much the data can be trusted given the reliability of its source. Regarding data Velocity, data can arrive and require processing at different speeds, While for some applications, the arrival and processing of data can be performed in batch, other analytics applications require continuous and real-time analyses, sometimes requiring immediate action upon processing of incoming data streams.[1].

Cyber Physical Systems [2]are integration of physical processes with computation and communication. Thus, Cyber Physical Systems connects the virtual world with the physical world. Cyber physical systems integrate physical devices, such as sensors and cameras, with cyber (or informational) components such as Cloud computing to form situation-integrated analytical system that responds intelligently to dynamic changes in the real-world scenarios. Cyber physical systems interconnect the cyber world with physical world by embedding sensors and computational nodes to the physical world. Applications of Cyber physical systems include, but are not limited to, automobile and intelligent transportation systems, aerospace systems, medical devices and health care systems, electric grid management, disaster recovery, factory automation, smart spaces, military applications, and environmental science research. Cyber physical systems not only have rosy visions but also face challenges. One of the key challenges is providing real-time data services for Cyber physical systems . Cyber physical systems have to deal with large amounts of data in a timely, secure fashion.

Big data driven cyber physical systems not only meet big data 4V feature [3] requirements, but also have to meet time constrains and spatial constraints of cyber physical systems. Big data driven cyber physical systems have to deal with time-constrained data and time-constrained transactions. In each of big data driven cyber physical applications, data about the target environment must be continuously collected from the physical world and processed in a timely manner to generate real-time responses [4]. Those systems contain a large network of sensors distributed across different components, which leads to a tremendous amount of measurement data available to system operators. The design of big data driven cyber physical systems requires the introduction of new concepts to model classical data structures, 4V features, time constraints and spatial constraints, and the dynamic continuous behavior of the physical world. In this paper, we propose a model based approach to model big data driven cyber physical systems based on integration of Modelica [5], Modelicaml [6] , AADL [7], RCC [8] and clock theory [9], we illustrate our approach by specifying and modeling Vehicular Ad hoc Networks (VANET) [10].

2 Challenges for Big Data Driven Cyber Physical Systems

In big data driven cyber physical system, its data is different from other types of data with the following aspects: Variety : there are continuous data in physical world and these data are described by differential-algebraic equations; these data have complex structure, there have not only mass of non-spatial data in cyber physical systems but also large number of spatial data. Spatial data have complex structure in many facets and high dimensional attributes space. Some physical data sources can produce staggering amounts of raw data. We must analyze the physical data and physical entities behavior because much of these data is of no use, and it can be filtered and compressed by orders of magnitude. Therefore, one challenge is to model physical entities and analyze the dynamic behavior and define these filters in such a way that they do not discard useful information. Volume: there are enormous amount of data in big data driven cyber physical systems. Spatio-temporal data is very large. Especial, the data of moving physical entities is increased in geometry growth over time. Spatio-temporal data captured through remote sensors are always big data. However, recent advances in instrumentation and sensor networks make the spatio-temporal data even bigger, and put several constraints on data computation and analytics capabilities. Velocity: physical entitles often have strict time constraints and require cyber systems react in real time. Especially, the operations of spatial data involve many spatial entities and spatial objects, each object or entity also contains a large number of points, lines or regions. Therefore, the operations of spatial data are more complex than ordinary large text and digital data.

Cyber physical system data processing challenges in real time are very complex. The four characteristics of big data are defined as volume, velocity, veracity and variety and Hadoop like system can processes the volume and variety part of it. In addition to the volume and variety, the cyber physical system must process the velocity of the data as well. And meeting the requirements of the velocity of big data in cyber physical systems is not an easy task. First, the system should be able to understand the physical data, acquire physical data, and collect the data generated by physical world and real time event stream coming in at a rate of millions of events per seconds. Second, it needs to handle the parallel or distributed processing of this data in real time as and when it is being collected. Third, it should perform real time data mining from moving data stream and spatial-temporal historic data. The big data driven cyber physical systems should be not also a low latency system so that the computation can happen very fast with near real time response capabilities, but also the big data driven cyber physical systems should meet spatial requirements when the physical objects move. A moving object represents the continuous evolution of a spatial object over time. Regarding big data modeling, an important question is how to represent a moving object. In contrast to static objects, moving objects [11]are difficult to represent and model. The efficiency of modeling methods for moving objects is highly affected by the chosen method to represent and analyze the continuous nature of the moving object. The data model defined for modeling continuously changing locations over time should be simple, extensive, though expressive, and be easy to use and implement.

In order to realize the full benefits of big spatio-temporal data, one has to overcome both computational and I/O challenges. We not only need new models that explicitly model spatial and temporal constraints efficiently, but further research is also required in the area of physical world modeling, moving object behavior analysis , moving object database, and model integration and transformation.

3 Big Data Driven Cyber Physical System Design

Big data driven cyber physical applications that require real-time processing of high-volume data streams are pushing the limits of traditional design methods. Big data driven cyber physical systems not only meet big data 4V feature requirements [12], but also have to meet time constrains and spatial constraints of cyber physical systems. Big data driven cyber physical systems have to deal with time-constrained data and time-constrained transactions. Especially, Spatio-temporal data models should enable the user to represent the dynamic behavior of moving objects over time. The dynamic behavior refers to the continuous change of the positions of moving objects over time. Spatio-temporal data models must have the capacity to specify moving points , moving lines and moving regions. Big data driven cyber physical system is hard to develop because developers need to consider functional properties, non-functional properties, such as timeliness, energy, memory, safety and reliability, dynamic continuous properties, spatial requirements and the interaction with physical world. The lack of specification and modeling methods and techniques has a very important impact, as the increasing complexity of cyber-physical systems built today pushes traditional development processes to their limits. In these development processes the different aspects and disciplines of mechanics, electrics, and software usually act isolated from each other, which inhibits taking advantage of the full potential of mechatronic solutions. One approach to overcome this separation of engineering disciplines is an integrated abstract model that serves as a common language for specification and analysis of the cyber physical system. By modeling and simulating the cyber physical system on an abstract level, design alternatives can be explored and a common interdisciplinary understanding of the physical world behaviors and the internal behaviors of the system can be fostered. In this paper, we propose a model based approach to model big data driven cyber physical systems based on integration of Modelica, Modelicaml, AADL, RCC and clock theory as shown in Fig.1.

Modelica and Modelicaml are used to model the physical world of aviation cyber physical systems, and AADL and Modelicaml are used to model cyber system. Modelica allows models to be defined in a declarative manner, modularly and hierarchically and various formalisms to be integrated in the more general Modelica formalism. The multi-domain modeling capability of Modelica allows integrating electrical, mechanical, hydraulic, thermodynamic, etc., model components within the same model of aviation cyber physical systems. The main aim of ModelicaML is to enable an efficient and effective way to use both Modelica and UML [13]/SysML [14] models reusing notations that are also used for software modeling. ModelicaML is based on a subset of the OMG Unified Modeling Language (UML) and reuses concepts from the OMG Systems Modeling Language (SysML). ModelicaML is designed towards the generation of Modelica code from graphical models. Since the ModelicaML profile is an extension of the UML meta-model it can be used for both: Modeling with standard UML and with SysML and Modelica .

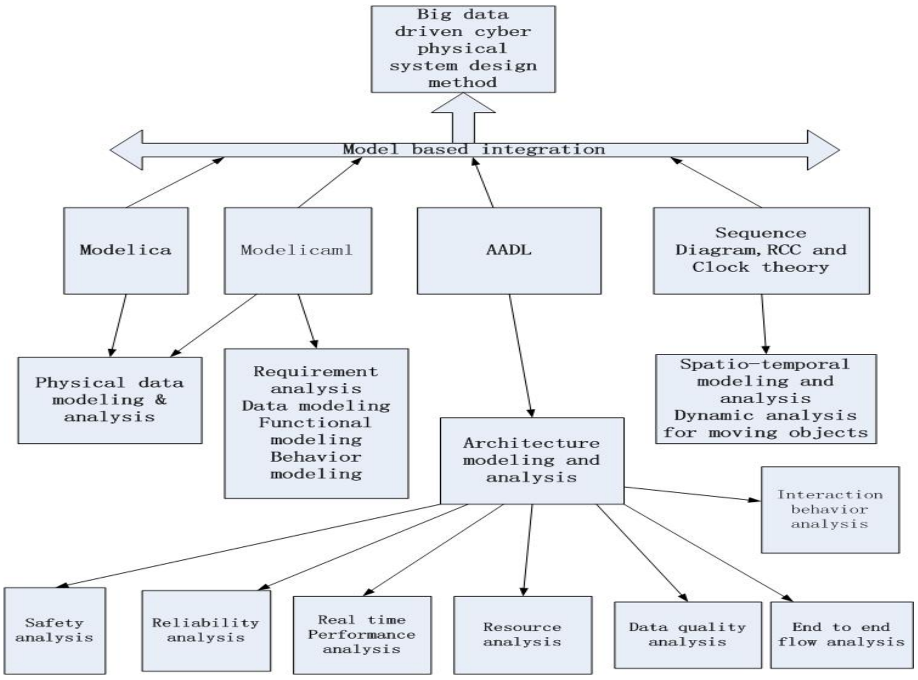

Big data driven cyber physical system design method
Model based integration
Modelica
Modelicaml
AADL
Sequence Diagram, RCC and Clock theory
Physical data modeling & analysis
Requirement analysis Data modeling Functional modeling Behavior modeling
Architecture modeling and analysis
Spatio-temporal modeling and analysis Dynamic analysis for moving objects
Interaction behavior analysis
Safety analysis
Reliability analysis
Real time Performance analysis
Resource analysis
Data quality analysis
End to end flow analysis

Fig. 1. Model based approach to model big data driven cyber physical systems based on integration of Modelica, Modelicaml, AADL, RCC and clock theory

Architecture Analysis & Design Language (AADL) is proposed by Society of Automotive Engineers (SAE). AADL provides the data component to model data types and data abstraction. We can use the data component to model the big data of cyber physical systems. The data component category supports representing data types and data abstractions in the source text at the appropriate level of abstraction for the modeling effort. The dat**a** *type* is used to type ports to specify subprogram parameter types. AADL defines Data Modeling Annex document [15] for the Architecture Analysis & Design Language v2.0 (AS5506A) to model complex data types [16].

The existing theory for specifying objects in both space and time is not applicable to Intelligent Transportation Systems for at least three reasons: (1) the theory ignores physical attributes and constraints on the objects; (2) the interaction of objects in space in time is not addressed; and (3) the range and precision of time resolution is not adequate for real-time spatio-temporal systems. The Spatio-temporal representation and reasoning methods should be powerful enough to express categories of motion that can be useful in a qualitative context, and be kept as simple as possible so that characterizing its properties is still possible in a precise way. Only then can we have a principled representation for motion that could unify the various needs for spatio-temporal representation and reasoning of Transportation Cyber Physical Systems. The expressive power of the combined spatio-temporal formalisms of the proposed spatio-temoral Sequence Diagram lays in the expressivity of the spatial requirements, the expressivity of the temporal requirement, and the interaction between the two components allowed in the combined logic. In this paper, we propose a spatio-temoral Sequence Diagram that integrates Region Connection Calculus RCC, clock theory and UML Sequence Diagram as shown in Fig.2, such a combined spatio-temporal formalism permits us to describe spatial configurations that change over time. we aim at specifying, modeling and analyzing some properties of space and time in a formalism which allows for the representation of relations between moving entities over time as shown in Fig.3.

A UML sequence diagram [17] illustrates a collaboration of interacting objects, where the interactions are invoked by exchange of messages. Its focus is on the temporal order of the message flow. Each object is assigned a column, the messages are shown as horizontal, labeled arrows, and a vertical time axis is assumed.

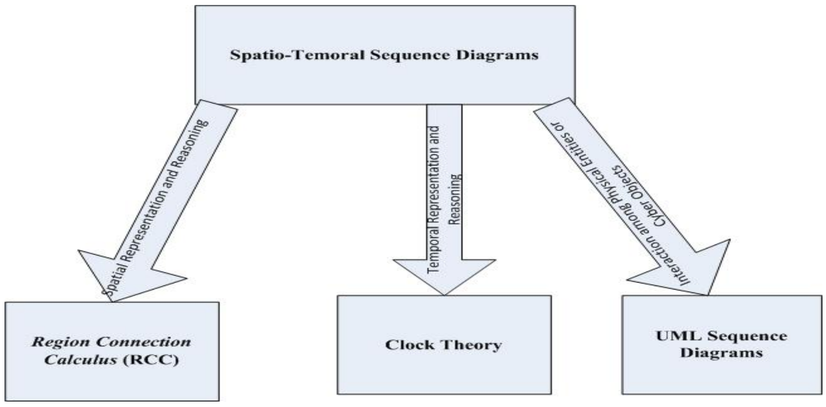

Spatio-Temoral Sequence Diagrams
Spatial Representation and Reasoning
Temporal Representation and Reasoning
Interaction among Physical Entities or Cyber Objects
Region Connection Calculus (RCC)
Clock Theory
UML Sequence Diagrams

Fig. 2. The Proposed Spatio-Temoral Sequence Diagram

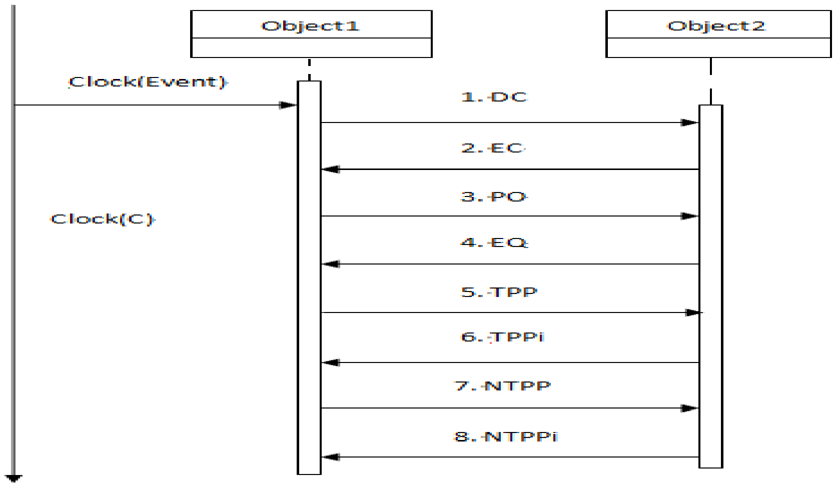


Fig. 3. The representation of spatio-temporal diagram in RCC-8 with the clock theory

RCC [18] is a logic theory for qualitative spatial representation and reasoning. The fundamental approach of RCC is that extended spatial entities, i.e.*regions* of space, are taken as primary rather than the dimensionless points of traditional geometry; and the primitive relation between regions — giving the language the ability to represent the structure of spatial entities— is that of *connection.* The RCC is an axiomatization of certain spatial concepts and relations in first order logic. The basic theory assumes just one primitive dyadic relation: C(x, y) read as "x connects with y". Individuals (x, y) can be interpreted as denoting spatial regions. The relation C(x, y) is reflexive and symmetric. Of the defined relations, Disconnected (DC), Externally Connected (EC), Partially Overlaps (PO), Equal (EQ), Tangential Proper Part (TPP), Non Tangential Proper Part (NTPP), Tangential Proper Part Inverse (TPPi) and Non Tangential Proper Part Inverse (NTPPi) have been proven to form a jointly exhaustive and pairwise disjoint set, which is known as RCC-8. Similar sets of one, two, three and five relations are known as RCC-1, RCC-2, RCC-3 and RCC-5. The syntax of RCC-8 [19] contains eight binary predicates,

• DC(X, Y) — regions X and Y are disconnected,
• EC(X, Y) — X and Y are externally connected,
• EQ(X, Y) — X and Y are equal,
• PO(X, Y) — X and Y partially overlap,
• TPP(X, Y) — X is a tangential proper part of Y ,
• NTPP(X, Y) — X is a nontangential proper part of Y ,
• the inverses of the last two—TPPi(X, Y) and NTPPi(X, Y),

The RCC-8 and RCC-5 relations between regions is shown in Fig.4.

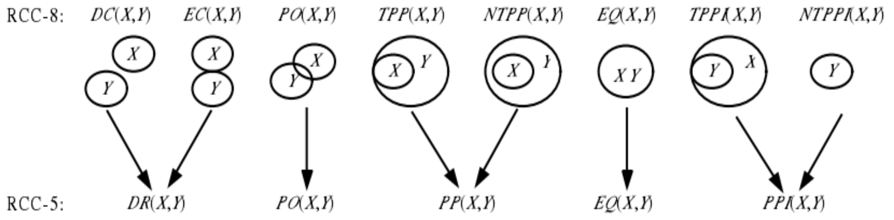


Fig. 4. RCC-8 and RCC-5 relations between regions

When modeling a cyber physical system not only structure and dynamic behavior have to be considered, but also timing constraints are mandatory for the correctness. In order to define relations changing through time and to recover some concepts of spatial (relative) localisation, we are going to define a concept of temporal part, called a temporal slice. A temporal slice of a spatio-temporal entity is the maximum part of a spatio-temporal region corresponding to the lifespan of another one.

Clock theory [20] puts forward the possibility to describe the event in physical world by using a clock, and can analyze, records the event by clock. To use clock to specify Cyber Physical Systems, the time description is clearer to every event and can link continuous world with discrete world better.

4 Case Study: Big Data Driven Vehicular Cyber Physical Systems Design

Vehicular Ad Hoc Networks (VANET) [21] captures, collect, possess and distribute traffic information to improve traffic congestion and to massively reduce the number of accidents by warning drivers about the danger before they actually face it. Vehicular Ad Hoc Networks (VANET) contain sensors and On Board Units (OBU) installed in the vehicle as well as Road Side Units (RSU). The data captures and collected from the sensors on the vehicles can be delivered and displayed to the driver, sent to the RSU and traffic control centers, or even broadcasted to other vehicles depending on its nature and importance. The traffic control center also can send traffic information to vehicles and RSU, control the states of vehicles and RSU, vehicles can communicate with other vehicles, The RSU distributes this data, along with data from road sensors, weather centers, traffic control centers, etc to the vehicles and also provides commercial services such as parking space booking, Internet access and gas payment.

Vehicular Ad hoc Networks (VANET) [22] are supported by a large amount of data that are captured and collected from various resources, are systems that would enable users to efficiently utilize data resources that pertain to intelligent transportation systems, access and employ data through more convenient and reliable services to improve the performance of transportation systems. The data In Vehicular Ad hoc Networks (VANET) include information, such as geographic space, location, speed, count, and behavior patterns, which are combined to obtain estimates of traffic conditions. As traffic data continues to grow, the average monthly data for traffic information has now reached 100 terabytes. The traffic volume, speed and occupancy data and position of vehicles over time have been regarded as important features in traffic control and information management systems such as vehicular Ad hoc Networks (VANET). For example, in an urban area with hundreds of thousands of vehicles, drivers and passengers in these vehicles want to get information relevant to their trip. They require that the location information van be displayed on screen and at any time, the available parking spaces around the current location of the vehicle are informed. The driver may be interested in the traffic conditions one mile ahead. Such information is important for drivers to optimize their travel, to alleviate traffic congestion, or to avoid wasteful driving.

There are enormous amount of data in Vehicular Ad hoc Networks (VANET). Spatio-temporal data for moving vehicles in Vehicular Ad hoc Networks (VANET) is very large. Especial, the data of moving vehicles is increased in geometry growth over time. Spatiote-temporal data captured through remote sensors in Vehicular Ad hoc Networks (VANET) are always big data.

The VANET includes ITS Vehicle station systems, IVS, ITS Roadside station system and ITS Control station system. The top Modelicalml [23] model of VANET is shown in Fig.5.

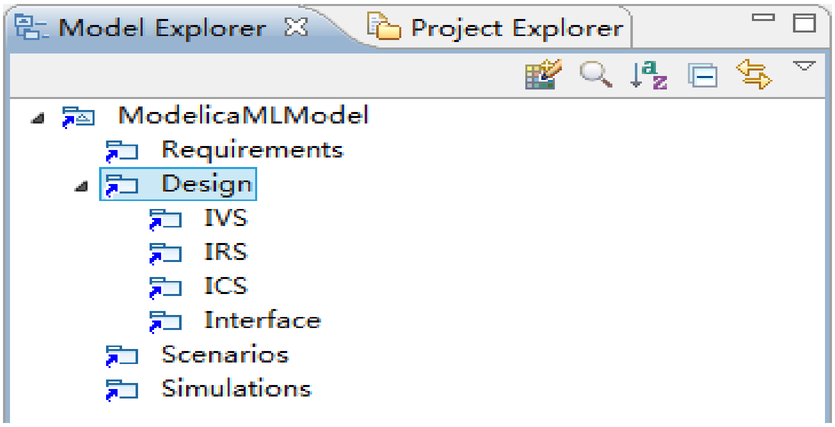


Fig. 5. . The top Modelicalml model of VANET

Fig.6 represents the requirements of VANET

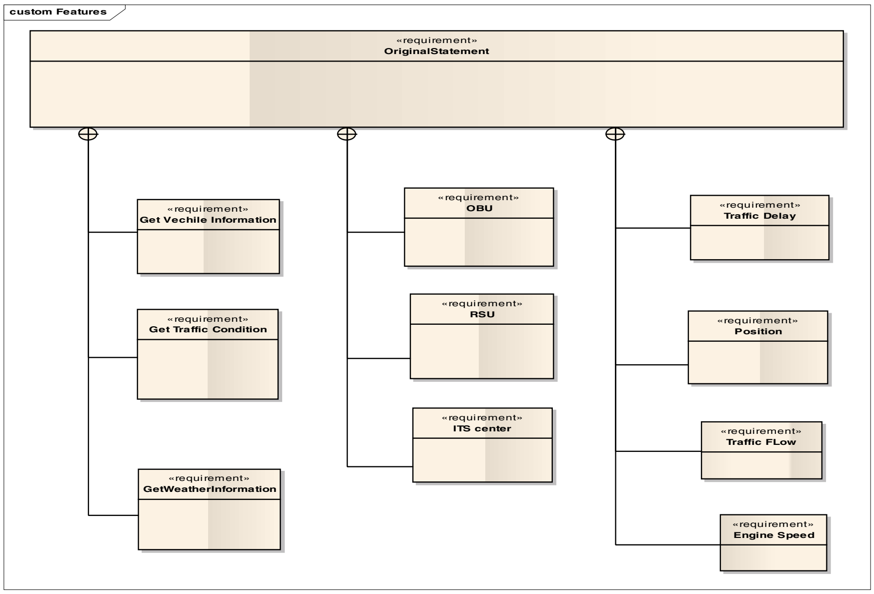

custom Features
«requirement» OriginalStatement
«requirement» Get Vechile Information
«requirement» Get Traffic Condition
«requirement» GetWeatherInformation
«requirement» OBU
«requirement» RSU
«requirement» ITS center
«requirement» Traffic Delay
«requirement» Position
«requirement» Traffic FLow
«requirement» Engine Speed

Fig. 6. The requirements of VANET

Fig.7 represents the Modelica model of vehicle.

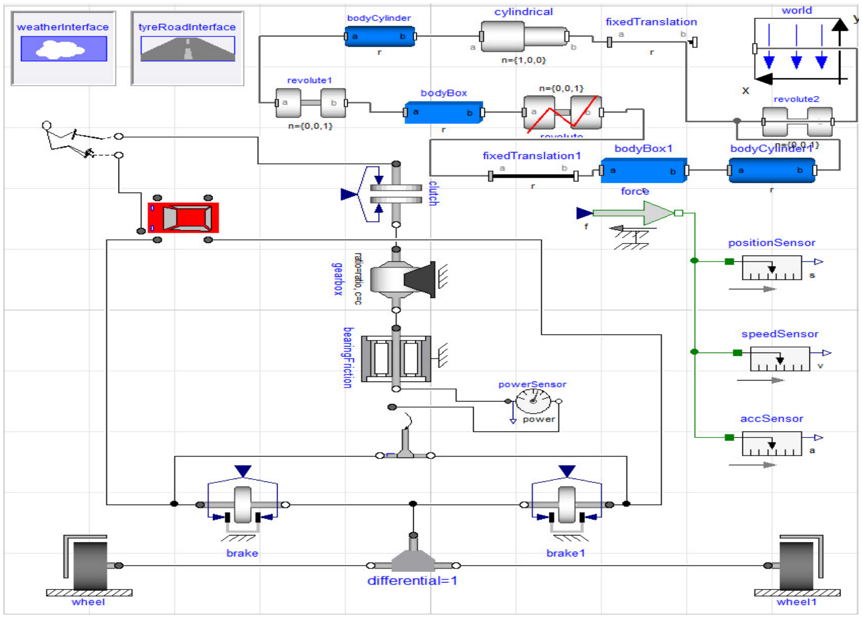


Fig. 7. The Modelica model of vehicle

Fig. 8 represents the use case diagram of traffic Control.

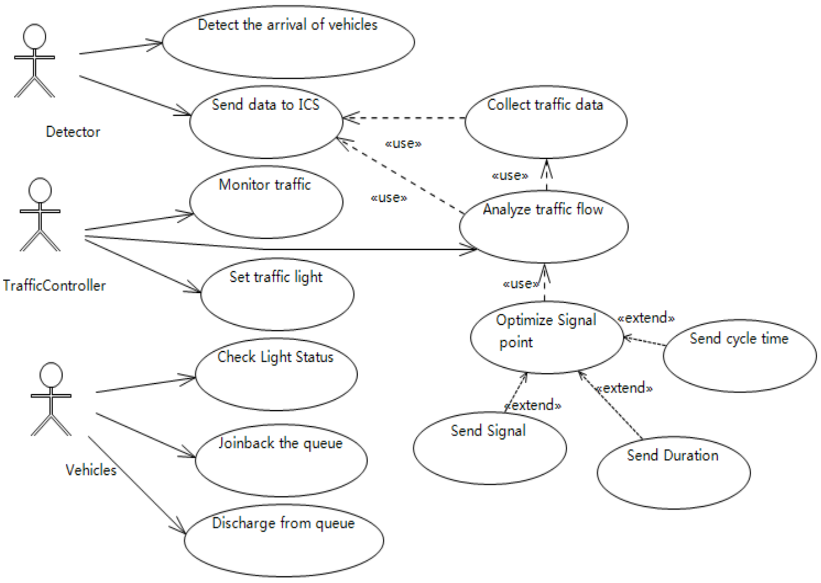


Fig. 8. The use case diagram of traffic Control

Fig. 9 represents the class diagram of traffic control

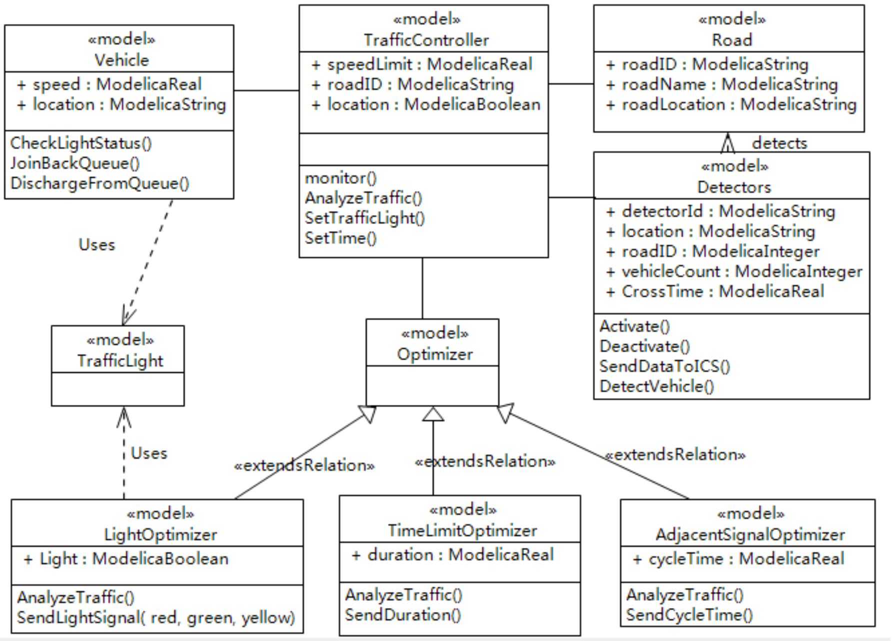

«model» Vehicle
+ speed : ModelicaReal
+ location : ModelicaString
CheckLightStatus()
JoinBackQueue()
DischargeFromQueue()
«model» TrafficController
+ speedLimit : ModelicaReal
+ roadID : ModelicaString
+ location : ModelicaBoolean
monitor()
AnalyzeTraffic()
SetTrafficLight()
SetTime()
«model» Road
+ roadID : ModelicaString
+ roadName : ModelicaString
+ roadLocation : ModelicaString
detects
«model» Detectors
+ detectorId : ModelicaString
+ location : ModelicaString
+ roadID : ModelicaInteger
+ vehicleCount : ModelicaInteger
+ CrossTime : ModelicaReal
Activate()
Deactivate()
SendDataToICS()
DetectVehicle()
Uses
«model» TrafficLight
«model» Optimizer
Uses
«extendsRelation»
«extendsRelation»
«extendsRelation»
«model» LightOptimizer
+ Light : ModelicaBoolean
AnalyzeTraffic()
SendLightSignal(red, green, yellow)
«model» TimeLimitOptimizer
+ duration : ModelicaReal
AnalyzeTraffic()
SendDuration()
«model» AdjacentSignalOptimizer
+ cycleTime : ModelicaReal
AnalyzeTraffic()
SendCycleTime()

Fig. 9. the class diagram of traffic control

Fig. 10 represents Connection Diagram for Traffic Light Control.

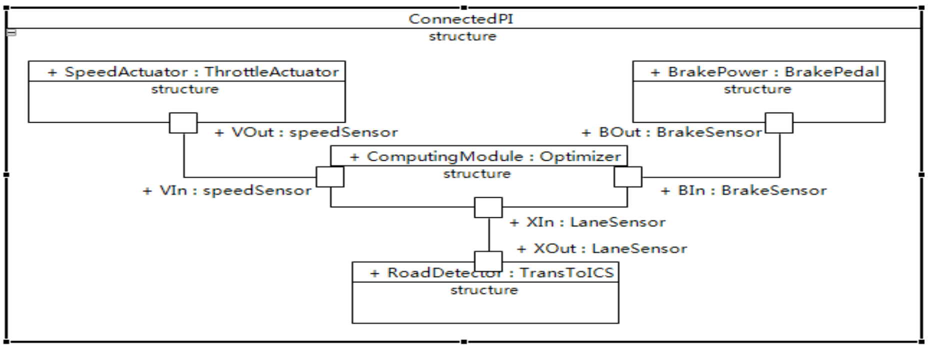


Fig. 10. Connection Diagram for Traffic Light Control

The Modelica model of the Optimizer in traffic light control is as follows:

```
within ModelicaMLModel.Scenarios.ICS;
model Optimizer
        Real S;
        output Real Ty;
```

```
          Real broom;
          Boolean safe;
          ModelicaMLModel.Scenarios.Interface.speedSensor VIn;
          ModelicaMLModel.Scenarios.Interface.BrakeSensor BIn;
          ModelicaMLModel.Scenarios.Interface.LaneSensor XIn;
          equation
          broom=VIn.V/2*BIn.B;
der(XIn.X)=VIn.V;
Ty = if (XIn.X/VIn.V>broom) then 0 else XIn.X/VIn.V;
S=VIn.V*VIn.V/2*BIn.B;
safe= if (XIn.X>S) then true else false;
end Optimizer;
within ModelicaMLModel.Scenarios.Interface;
connector BrakeSensor
          parameter input Real B=25;
  end BrakeSensor;
within ModelicaMLModel.Scenarios.Interface;
connector speedSensor
          constant input Real V=13;
end speedSensor;
within ModelicaMLModel.Scenarios.Interface;
connector LaneSensor
          Real X;
end LaneSensor;
model ConnectedPI
          ModelicaMLModel.Scenarios.ICS.Optimizer ComputingModule;
          ModelicaMLModel.Scenarios.IVS.ThrottleActuator SpeedActuator;
          ModelicaMLModel.Scenarios.IVS.BrakePedal BrakePower;
          ModelicaMLModel.Scenarios.IRS.TransToICS RoadDetector;
equation
          connect(SpeedActuator.VOut, ComputingModule.VIn);
          connect(BrakePower.BOut, ComputingModule.BIn);
          connect(ComputingModule.XIn, RoadDetector.XOut);
end ConnectedPI;
```

When we specify spatio-temporal relations of vehicle to vehicle using Spatio-Temporal Sequence Diagram, we need to walk a fine line between realism and a suitable level of abstraction. Since automobile driving is a physically complex process, realistic models are often also complex. We define four basic operations: Acc:accelerate, Dec: decelerate, Trans: transform lanes, Turn: turn lanes, Uniform :constant.

Fig.11 represents the car spatial relations in cross road.

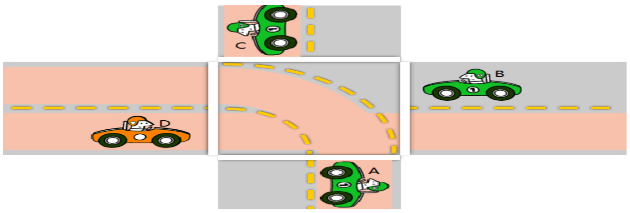


Fig. 11. The car spatial relations in cross road

Fig. 12. represents the spatio-temporal sequence diagram of Cross roads.

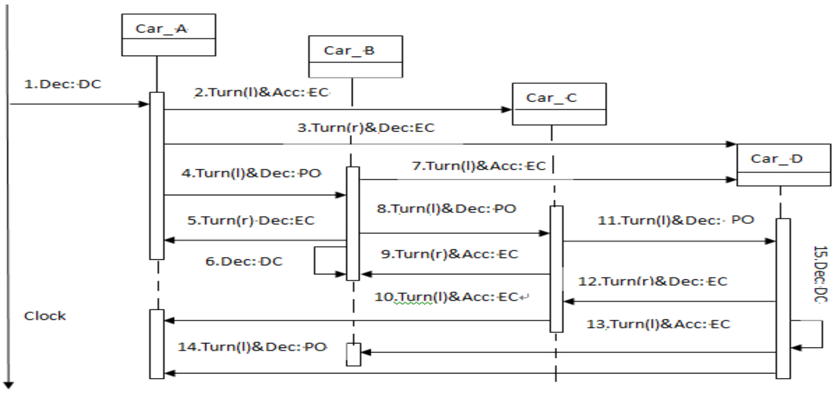


Fig. 12. The spatio-temporal sequence diagram of Cross roads

clock(Turn) $\preceq$ clock(Uniform) $\preceq$ climb(Car_B, Car_A) $\preceq$ clock(Dec) $\preceq$ clock(Acc)) $\preceq$ climb(Car_A, Car_C) $\preceq$ clock(Uniform) $\preceq$ clock(Turn) $\preceq$ clock(Dec) $\preceq$ clock(Acc) $\preceq$ drop(Car_A, Car_B)

ρ(clock(Turn), clock(Uniform)) ≤ d_{AB}/V_B
ρ(clock(Uniform), climb(Car_B, Car_A)) ≤ d_{AB}/V_B
ρ(climb(Car_B, Car_A), clock(Dec)) ≤d_{AB}/V_B
ρ(clock(Dec), clock(Acc)) ≤d_{AB}/V_B
ρ(clock(Acc), climb(Car_A, Car_C)) ≤d_{AC}/V_A
ρ(climb(Car_A, Car_C), clock(Uniform)) ≤d_{AC}/V_A
ρ(clock(Uniform), clock(Turn)) ≤d_{AC}/V_A
ρ(clock(Turn), clock(Dec)) ≤d_{AC}/V_A
ρ(clock(Dec), clock(Acc)) ≤d_{AC}/V_A
ρ(clock(Acc), drop(Car_A, Car_B)) ≤d_{AB}/V_B

clock(Dec) $\preceq$ clock(Uniform) $\preceq$ clock(Turn) $\preceq$ climb(Car_A, Car_B) $\preceq$ clock(Acc)) $\preceq$ drop(Car_A, Car_D)

ρ(clock(Dec), clock(Uniform)) $\leq d_{AB}/V_A$
ρ(clock(Uniform), clock(Turn)) $\leq d_{AB}/V_A$
ρ(clock(Turn), climb(Car_A, Car_B)) $\leq d_{AB}/V_A$
ρ(climb(Car_A, Car_B), clock(Acc)) $\leq d_{AB}/V_A$
ρ(clock(Acc), drop(Car_A, Car_B)) $\leq d_{AB}/V_B$

5 Conclusion

Big data driven cyber physical systems not only meet big data 4V feature requirements, but also have to meet time constrains and spatial constraints of cyber physical systems. Regarding big data modeling, an important question is how to represent a moving object. In this paper, we propose a model based approach to model big data driven cyber physical systems based on integration of Modelica, Modelicaml, AADL, RCC and clock theory, we illustrate our approach by specifying and modeling Vehicular Ad hoc Networks (VANET). The main advantages of the proposed approach is its capacity to take into account big data properties and cyber physical system properties through specialized concepts in rigorous, easy and expressive manner.

In future work , we extend MapReduce in real time aspects to of enable the scheduling of mixed hard and soft real-time MapReduce applications, we extend our method to model big data driven cyber physical systems on cloud platforms and SOA.

Acknowledgments. This work is supported by the national natural science foundation of China under grant (No.61370082, No.61173046, No.91318301)), natural science foundation of Guangdong province under grant (No.S2011010004905). This work is also supported by Shanghai Knowledge Service Platform Project (No.ZF1213)

References

1. Assuncao, M.D., Calheirosb, R.N., Bianchia, S., Netto, M.A.S., Buyya, R.: Big Data Computing and Clouds: Challenges, Solutions, and Future Directions. Technical Report CLOUDS-TR-2013-1, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne (2013)
2. Eidson, J., Lee, E.A., Matic, S., Seshia, S.A., Zou, J.: Distributed Real-Time Software for Cyber-Physical Systems. Proceedings of the IEEE (special issue on CPS) 100(1), 45–59 (2012)
3. Kaisler, S., Armour, F., Espinosa, J.A., Money, W.: Big Data: Issues and Challenges Moving Forward. In: International Conference on System Sciences, pp. 995–1004. IEEE Computer Soceity, Hawaii (2013)
4. Don, S., Min, D.: Medical Cyber Physical Systems and Bigdata Platforms. In: The Fourth Workshop on Medical Cyber-Physical Systems (2013)

5. Mattsson, S.E., Elmqvist, H., Otter, M.: Physical system modeling with Modelica. Control Engineering Practice 6, 501–510 (1998)
6. Schamai, W.: Modelica Modeling Language (ModelicaML) A UML Profile for Modelica. technical report, Germany, Linkoping University, Sweden (2009)
7. SAE: Architecture Analysis & Design Language (AADL) v1, AS5506. SAE Aerospace Standard (2004)
8. Randell, D., Cui, Z., Cohn, A.: A spatial logic based on regions and connection. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, pp. 165–176 (1992)
9. Jifeng, H.: A Clock-Based Framework for Construction of Hybrid Systems. In: Liu, Z., Woodcock, J., Zhu, H. (eds.) ICTAC 2013. LNCS, vol. 8049, pp. 22–41. Springer, Heidelberg (2013)
10. Zeadally, S., et al.: Vehicular ad hoc networks (VANETS): status, results, and challenges. Telecommunication Systems 50(4), 217–241 (2012)
11. Vazirgiannis, M.: Son. W.: A spatiotemporal model and language for movign objects on road networks. In: MOBIDE (2001)
12. Bajaj, R.H., Ramteke, P.L.: Big Data – The New Era of Data. International Journal of Computer Science and Information Technologies 5(2), 1875–1885 (2014)
13. OMG: OMG Unified Modeling Language TM (OMG UML) Version 2.2 (2009)
14. OMG: OMG Systems Modeling Language (OMG SysML^{TM}) Version 1.1 (2008)
15. SAE/AS2-C: Data Modeling Annex document for the Architecture Analysis & Design Language v2.0, AS5506A (2009)
16. Gilles, O., Hugues, J.: Expressing and enforcing user-defined constraints of AADL models. In: Proceedings of the 5th UML and AADL Workshop (2010)
17. Seemann, J., von Gudenberg, J.W.: Extension of UML Sequence Diagrams for Real-Time Systems. In: Bézivin, J., Muller, P.-A. (eds.) UML 1998. LNCS, vol. 1618, pp. 240–252. Springer, Heidelberg (1999)
18. Serrano, M.A., Romero, J.G., Patricio, M.A., García, J., Molina, J.M.: Applying the Dynamic Region Connection Calculus to exploit geographic knowledge in maritime surveillance. In: FUSION, pp. 1546–1553 (2012)
19. Griffiths, A.A.: Computational Properties of Spatial Logic in the Real Plane. Ph.D thesis, the University of Manchester (2008)
20. Xu, B., He, J., Zhang, L.: Specification of Cyber Physical Systems Based on Clock Theory. International Journal of Hybrid Information Technology 6(3), 45–54 (2013)
21. Adam, H.S., Gmbh, O., Simtd, B.M.: A Car-to-X System Architecture for Field Operational Tests. IEEE Communications Magazine 48(5), 148–154 (2010)
22. Yousefi, S., Mousavi, M.S., Fathy, M.: Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives. In: Proceedings of 6th International Conference on ITS Telecommunications, pp. 761–766 (2006)
23. Wladimir, S., Fritzson, S.P., Paredis, C., Pop, A.: Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. In: Proceedings 7th Modelica Conference (2009)

Reliable Transmission with Multipath and Redundancy for Wireless Mesh Networks

Wenze Shi[1,2], Takeshi Ikenaga[2], Daiki Nobayashi[2], Xinchun Yin[1], and Yebin Xu[1]

[1] School of Information Engineering, Yangzhou University, Yangzhou, China
swz4123@qq.com
[2] Graduate School of Engineering, Kyushu Institute of Technology, Fukuoka, Japan
{ike,nova}@ecs.kyutech.ac.jp

Abstract. Wireless Mesh Networks (WMNs) is considered as a key technology to solve the last mile problem. However due to the nature weakness of wireless channels, the packet transmission in WMNs is not that reliable. In this paper we proposed a new packets transmission scheme which is called – Reliable Transmission with Redundancy and Multipath (RTRM). RTRM uses ETT as its routing metric which not only consider the Packets Delivery Ratio (PDR) but also consider bandwidth. In RTRM we use scalar redundancy strategies and will choose a proper redundancy strategy depends on the minimum ETT of all paths. What's more, how to send redundant packets will be considered. We compared four different strategies to send redundant packets and add the best one into RTRM. By using RTRM the WMNs will have more capacity of fault-tolerance and can offer reliable Quality of Service (QoS) guarantee.

Keywords: wireless mesh network, redundant packets, multipath routing, reliable transmission, scalar redundancy, QoS.

1 Introduction

Wireless Mesh Networks (WMNs) is a new kind of broadband wireless access network which aims at providing networks anytime anywhere [1]. In WMNs these nodes can dynamically self-organize and self-configure and are able to establish and maintain mesh connectivity automatically. What's more, other wireless networks, like Ad hoc, focus more on mobility while WMNs cares more about wireless. Most of the base stations in WMNs have low mobility. All these characteristics make WMNs reliable and flexible. However it's not enough since wireless channels have high noise, fading and easy to suffer interference which can cause packets loss during transmission. When the distance between two radio nodes becomes larger, the Signal-to-Noise decreases [2], the packets loss rate will be highly increased. Our target in this paper is to improve the fault-tolerant ability of WMNs and extend its reliable transmission radius. Multipath routing will be adopted because if one path failed, we don't need to recovery that path since other paths still work. What's more, multipath routing can distribute network traffic to ease up the load of each path. Scalable redundancy strategies are also essential since it can balance the number of redundant packets and the ability of recovering lost packets.

The rest of this paper is organized as follows. Section 2 introduces some related works. Section 3 shows how to scale the redundant strategy. In Section 4, we give the implementation of Reliable Transmission with Redundancy and Multipath (RTRM). Section 5 shows the performance evaluations of our scheme. Finally, conclusion and future works are given in Section 6.

2 Related Works

Fault-tolerance on networks has been studied for decades. Those proposed schemes is divided into two categories: hardware level and software level. The software level contains network coding, adding redundant packets, multipath routing, data fusion.

Multipath routing combines with packet redundancy is a main method to enhance the fault-tolerant capacity [3]. Lei Wang et al. proposed a Multipath Source Routing (MSR) which uses two paths to transmit original packets and duplicate packets respectively to guarantee the reliability [4]. However, in MSR duplicate packets will add 100% of extra traffic into network. And their metric RTT cannot estimate the packet loss rate of a path , so that they may not choose a proper path.

Akyildiz et al. proposed a Forward Error Correction (FEC) based schemes for underwater sensor networks [5]. This kind of networks usually has low bandwidth, large propagation delay, high error rate, half-duplex channels, and highly dynamic topology. They designed a reliable transmission protocol by adding redundant information into original data. However the difficulty is to decide how many redundant packets should be added.

Network coding is another popular method to enhance network reliability [9].Lun et al. use random linear network coding to construct a capacity-approaching scheme. But their algorithm does not guarantee 100% Packets Delivery Ratio (PDR) [6]. Zhenyu Yang et al. proposed a R-Code network coding based reliable broadcast protocol [7] which can achieve 100% PDR. Nevertheless they didn't give the reliable transmission radius for the PDR cannot always be 100% when the distance between two nodes extend.

Adding redundant packets is surely able to make the packets transmission more reliable. However this kind of resource hungry method would increase 100% of network load. In this paper we aimed at finding out a best tradeoff between redundancy and reliability. Our reliable transmission scheme RTRM adopts XOR operation as its encoding method. In this case we can rank the redundancy in different levels. Heavy redundancy strategy will add more redundant packets and maybe get better performance. Therefore, how to choose a proper redundancy strategy becomes the key point. What's more, how to send redundant packet in multipath will influence the performance of RTRM. We will proof round robin is most suitable for RTRM.

3 Scalable Redundancy Strategy

In RTRM we designed a scalable redundancy strategy which uses XOR operation to generate redundant packets.

3.1 XOR Operation

Since the packet transmission in WMNs is not reliable, we try to add redundant packets to increase the PDR. Theoretically, more redundant packets leads to higher PDR. However, too much redundancy will consume a large amount of network resources and even cause network congestion. Therefore, how to tradeoff between redundancy and reliability becomes a key research point.

Here we adopt XOR operation as redundant packets generating method for it is a popular way of network coding and easy to implement [10]. It not only can generate redundant packets but also is scalable due to its excellent feature. We know that if $a \oplus b = c$ then $a \oplus c = b$ and $b \oplus c = a$. That means if any element in a and b loses, it can be recovered by doing XOR between the other element and c. This feature can be extended like, if $a_1 \oplus a_2 \oplus \cdots \oplus a_n = c$ then $a_1 \oplus ... \oplus a_{i-1} \oplus a_{i+1} \oplus ... \oplus a_n \oplus c = a_i \quad (1 \le i \le n)$. That means if any one of these n elements loses, it can be recovered by doing XOR among c and other elements. We define the redundancy rate R equals to formula (1):

$$R = \frac{1}{n} * 100\% \tag{1}$$

From formula (1) we see that R is controllable by changing the value of n. What's more this feature of XOR is suitable when these elements are changed to packets which have the same size. For example, when we use duplicate as redundant packets, the value of n equals to 1, then R equals to 100%.

3.2 Redundancy Strategy

When n is small, that means the R is large, and the reliability would be higher theoretically. Meanwhile the network should undertake heavier extra load. When the value of n becomes larger, extra load of network will be smaller, but the reliability may decrease. In order to choose a proper value of n, we did some simulations in a two nodes single path scenario by adding redundant packets to get the relationship between PDR and Distance. The topology is like Fig. 2(a) shows. The value of n varies from 0 to 10. When n equals to 0 that means no redundant packet will be added. Table 1 shows the parameters of this scenario.

In order to make the redundant strategy more reasonable, we rank the redundancy in 5 scales. Table 2 shows these 5 redundancy strategies and some related parameters like n, *Redundancy rate*, *Delivery threshold*. Here *Delivery threshold* means without redundancy if the PDR of one path is larger than this threshold, we can use the corresponding strategy to increase the PDR up to 99%. Fig.1 shows the PDR to Distance in 5 redundancy scales.

Table 1. Important parameters of scenario

Name of Parameter	Value
Wireless standard	802.11a
Bandwidth	48Mbps
CBR	1Mbps
Number of nodes	2

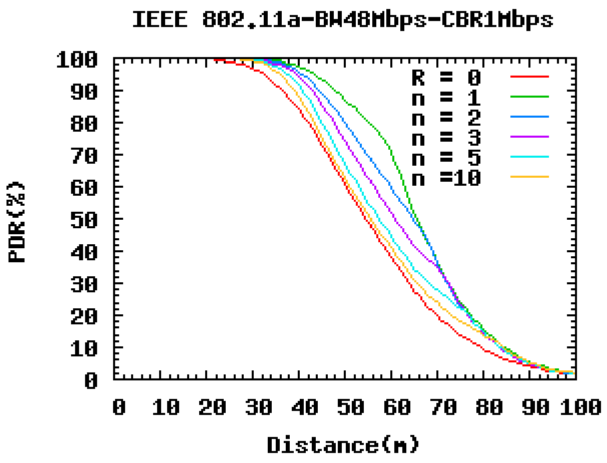


Fig. 1. PDR to Distance in 5 redundancy scales when Bandwidth is 48Mbps

Table 2. Details of 5 redundancy strategies

Scales	*n*	*Redundancy rate* (%)	*Delivery threshold* (%)
1	1	100	90
2	2	50	91.5
3	3	33	93
4	5	20	95
5	10	10	96.5

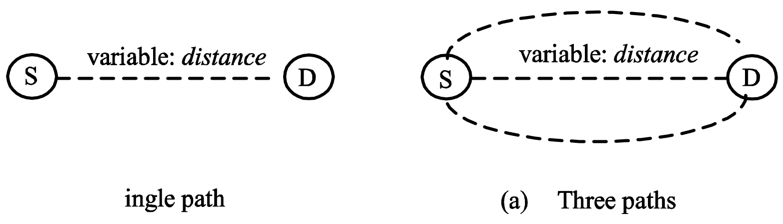


Fig. 2. Topology of simulation scenarios

4 Implementation

In RTRM we decide to use Multipath and Source Routing based on the following 3 reasons.

- **Risk-sharing**: Wireless channels have high noise, fast fading and easy to suffer interference. So we need multipath to distribute network traffic [11].
- **Weak mobility**: Wireless mesh network focuses more on wireless than mobility. At most of the time the base stations are fixed. That means their topologies are comparatively stable.
- **Path measurement**: If we use source routing, it would be easier to measure the metrics of these paths. Thus the redundancy scheme could be predetermined.

4.1 Multipath Routing

Simulation was hold to compare the reliability between a two node single path scenario and a two node three paths scenario which are shown in Fig. 2(a) and Fig. 2 (b). The CBR of source node is 12Mbps which is high data rate. Other parameters are the same as Table 1 shows.

The simulation result is given in Fig. 3. It shows that adopt multipath looks much more reliable than single path since its PDR is much higher than single path. However, actually the enhancement of reliability by using multipath is not that much, because most of the time its PDR is close to 99% but below 99%. Nevertheless, we believe that combine multipath and redundancy while transmission will satisfy our requirements of reliability.

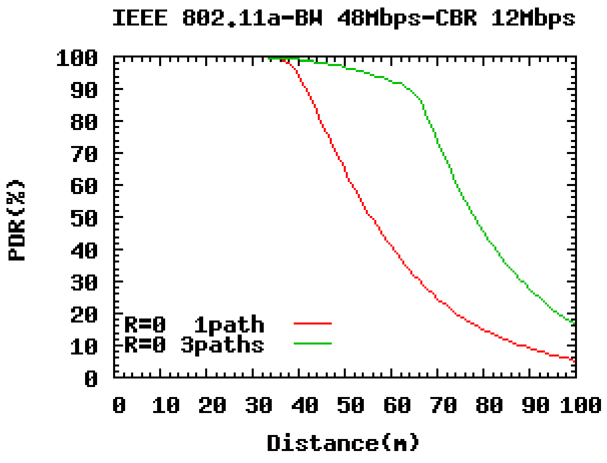


Fig. 3. PDR to Distance in comparison of single path and multipath

4.2 Route Discovery

There are many existent multipath source routing protocols, our research will not focus on how to find paths. The route discovery phase is briefly as follows:

Step 1: When the source node has sent demand, it broadcasts route discovery packets neighbors. Each route discovery packet contains a route vector which is used to recode the route it has passed.

Step 2: If a intermediate node receives a route discovery packet, firstly, it will check if itself has already been in the route vector. If so, it will drop this discovery packet immediately. Otherwise it will add itself into the route vector. This step can prevent routing loop and reduce the number of probing packets.

Step 3: When a route discovery packet gets to the destination node, it will be sent back from the reverse direction of its route vector.

Step 4: Source node receives many available route vectors, it can measure the metric of each path. Here we use ETT as the path metric because it not only consider the packet loss rate but also take bandwidth into account. Both of these two factors are essential when deciding redundancy strategy.

Step 5: Then we choose at most 3 paths from the best to the worst. When choose one path, we should compare its path vector with those we have chosen. This step can ensure these paths are node disjoint.

4.3 Implementation of RTRM

Fig.4 shows the network model of RTRM. Since IEEE 802.11 standard has already defined a relatively reliable MAC layer, we put our attention between Network layer and Data Link layer. Between these two layers we add a pair of Redundant Packet Generator (RPG) and Redundant Packets Filter (RPF).

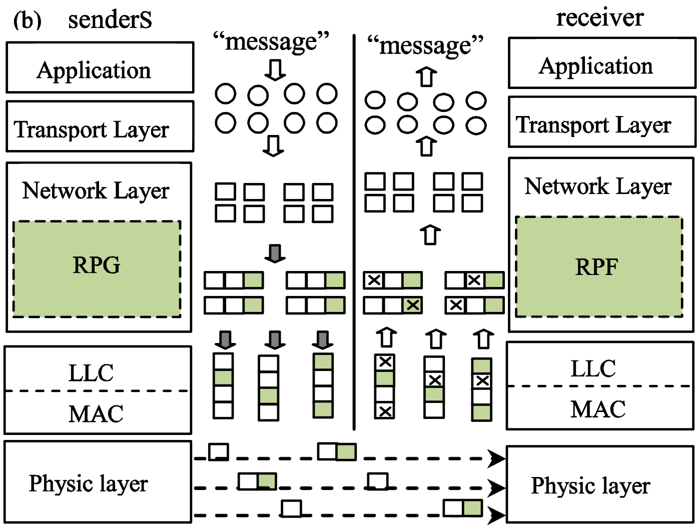


Fig. 4. Network model of RTRM

Functions of RPG:

- Generate redundant packets according to the redundancy strategy decided by ETT.
- Distribute network traffic.

Functions of RPF:

- Recover lost original packets.
- Filter out redundant packets.

The implementation of RTRM is as follows:

Step1: According to the metric ETT of selected paths, we can choose a proper redundancy strategy. Then the value of *n* is decided.

Step2: All packets through the network layer which have a same destination address will be allocated a same FlowID and a different SeqID. FlowID shows these packets are in a same flow and SeqID represents the sequence of a packet in that flow.

Step3: When original packets go through RPG, RPG will classify those packets which have the same FlowID into groups, each group includes *n* original packets.

Step 4: RPG makes all packets inside a group do XOR operation and get a redundant packet. That redundant packet will get a same GroupID and FlowID. What's more it will have a redundant flag to sign it is a redundant packet.

Step5: RPG allocates each packet a PathID. Original packet will be transmitted by round robin, while redundant packets will follow some redundant packet transmission strategies. We design 4 redundant packet transmission strategies and will compare them latter.

Step6: In RPF, there is a two-dimensional structure array, and the subscript is FlowID and GroupID. Each structure includes 4 elements:

- The number of group packets *n*.
- A *counter* to calculate how many packets in the same group has been received.
- A *recover* which has the same size as original packet. Its initialization value is 0.
- A *flag* which is the sign of redundant packet.

When any packet arrives, it can find a particular structure for its group. RPF will make every arrived packet and the *recover* do OXR operation and recode the value in the *recover*. After that if the packet is a redundant packet, RPF will filter it out. Otherwise, RPF will submit it to upper layer.

Step7: the transmission results of a group can be divided into the following 3 conditions:

- If the value of *counter* of a particular equals to n+1, that means all packets are delivered.
- If the value of *counter* equals to *n*, then we check the value of *flag*. If *flag* equals to 1, that means 1 original packet is lost. Then we submit recover to upper layer as the lost packet.
- If the value of *counter* is littler than *n*, that means in a group more than 1 original packet has been lost. In this case we can do nothing.

Redundant Packet Transmission Strategy

In order to preserve the good put while enhance reliability, we decide to transmit redundant packets during the interval of two original packets. Experiments show that, at what time during the interval to send redundant packet is indifferent. However in what way to send redundant packets lead to different results. We designed 4 redundant packet transmission strategies and described as follows:

- **Follow the same path** (same): Use the same path as the latest original packets.
- **Use next path** (next): Use the path which will be used by next original packet.
- **Use last path** (last): Use the path which has been used before the latest original packets.
- **Round robin**: Treat redundant packet as original packet and send them by round robin.

Fig.5 shows the original and redundant packets sequence by these 4 different redundant packet transmission strategies when *n* equals to 2. Three lines represent three paths in time axis.

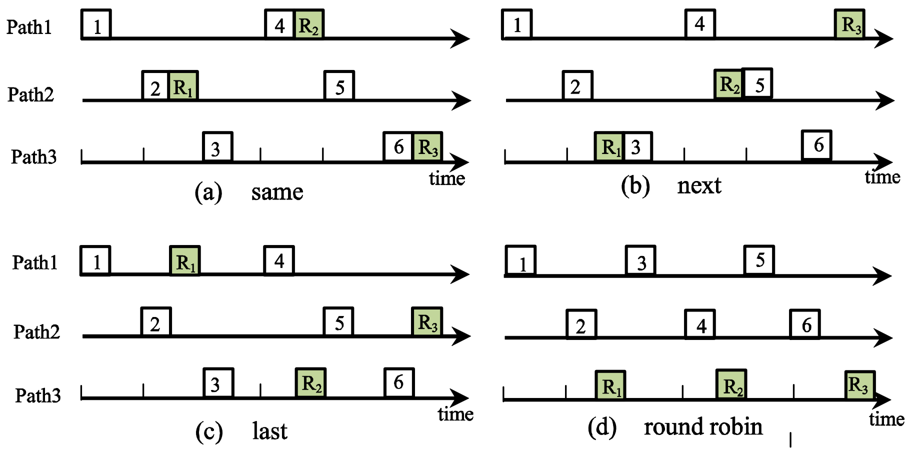


Fig. 5. Original and redundant packets sequence by different strategies

5 Performance Evaluations

Our simulation was held on Qualnet simulator [8]. There are two scenarios, one is low data rate scenario, the other one is high data rate. And the topology is like Fig.2 shows, one is two nodes single path topology, the other is two nodes three paths topology. In both scenarios, distance is a variable which varies from 1 meter to 100 meters.

5.1 Under Low Data Rate

In low data rate scenario the CBR of source node is 1Mbps.

We compare the performance of RTRM in 5 different redundancy strategies and the condition without redundancy when bandwidth equals to 12Mbps,18Mbps, 24Mbps, 36Mbps and 54Mbps. The topology is two nodes single path. Their results are just the same as Fig.1 shows. From the result we get that under low data rate higher redundancy can achieve better performance. This conclusion can be applied to all bandwidths.

5.2 Under High Data Rate

In high data rate scenario, the CBR of source node is 12Mbps.

Single Path Topology

Here we still use single path with redundancy. The result is like Fig.6 shows. From this figure we can see that under high data rate heavier redundancy strategy even has worse reliability. Dramatically, transmission without redundancy has the best performance.

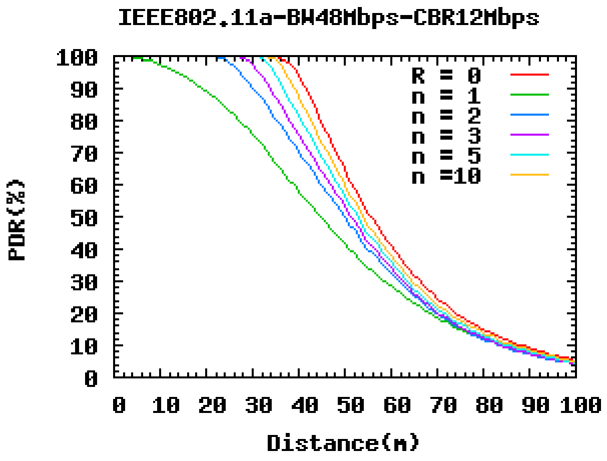


Fig. 6. PDR to Distance under high data rate when Bandwidth is 48Mbps

Multipath Topology

In this scenario, we adopt three paths topology with 4 different redundant packet transmission strategies. Fig.7, Fig.8, Fig.9 and Fig.10 show their results respectively. In these figures the red curve represents no redundancy on multipath.

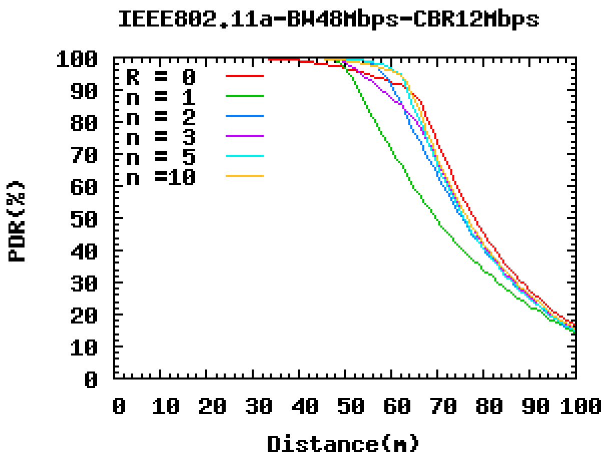


Fig. 7. PDR to Distance use “same” as redundant packet transmission strategy

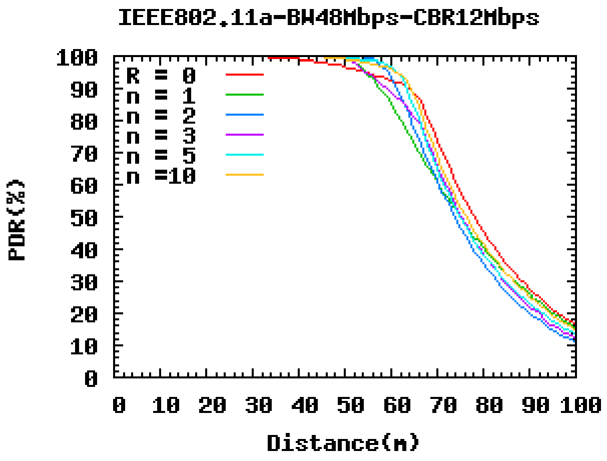


Fig. 8. PDR to Distance use “next” as redundant packet transmission strategy

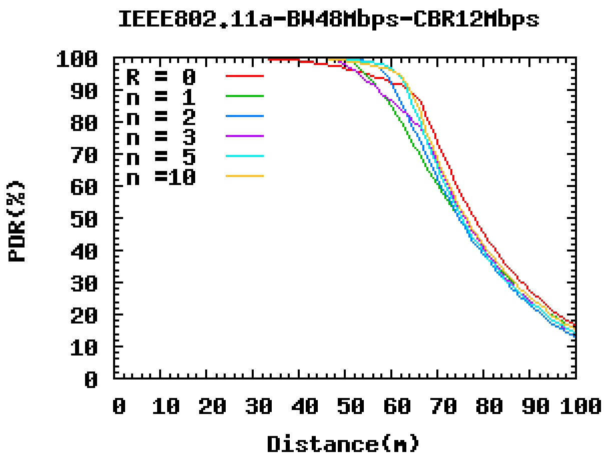


Fig. 9. PDR to Distance use "last" as redundant packet transmission strategy

From these figures we can briefly see that transmission with redundancy can provide better reliability. However, heavier redundancy strategies are not always better than lighter redundancy strategies. Fig.11 quantitatively compares the reliable radius of different redundancy scales and different redundant packets transmission strategies. Here we consider that where PDR is higher than 99% is reliable area.

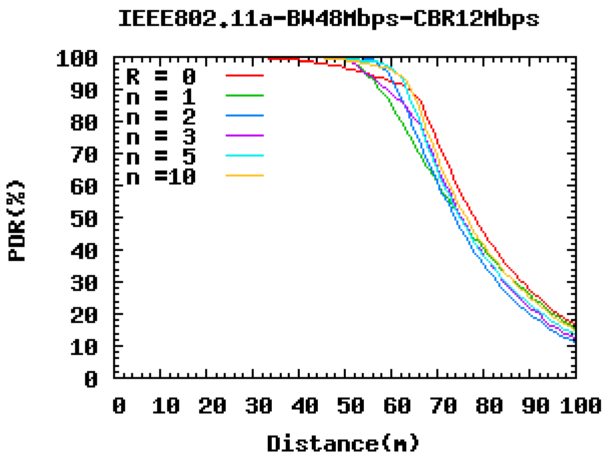


Fig. 10. PDR to Distance use "round robin" as redundancy transmission strategy

From this Fig.11 we can get three conclusions:

- Different redundant packets transmission strategy can affect the performance of RTRM especially in some particular redundancy strategies.
- Round robin achieves the best performance compares with other redundant packet transmission scheme.
- Redundancy strategy scale 4 provides the best reliability and its redundancy rate is only 20%.

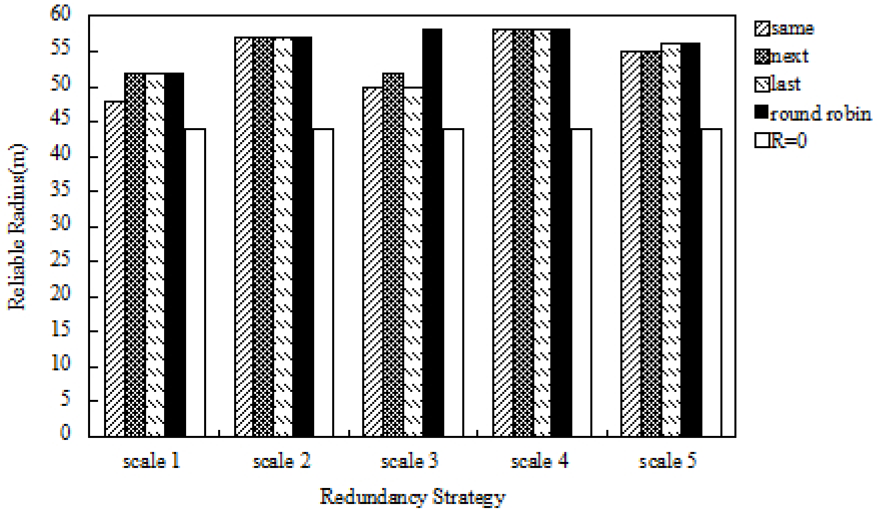


Fig. 11. Performance evaluations of different redundancy strategies

5.3 Discussion

From the simulation results, we can see that under low data rate scenario, higher redundancy rate can get better reliability. However under high data rate scenario, heavy redundancy strategy may add too much extra load while the enhancement of reliability is not desirable. We draw the conclusion that RTRM is sensitive to data rate. Under high data rate scenario we cannot choose redundancy strategy according to the value of ETT. High redundancy may cause the reliability worse. Here we recommend to use RTRM with round robin as its redundant packet transmission strategy and to set the redundancy strategy on scale 4 or scale 3for it can prove the best reliability and inject least redundant packets.

6 Conclusion and Future Works

In this paper we proposed a new packet transmission scheme named RTRM. By adding redundant packets and using multipath, the reliability of WMN is significantly enhanced. And we prove that Round robin is the best redundant packet transmission scheme. The most important thing is we find out that high redundancy is not always effectual in enhancing the transmission reliability. To use scale 3 or scale 4 maybe a better choice.

Our future works will focus on finding out the relation between data rate and reliability, since we find that wireless channel is sensitive to data rate.

Acknowledgement. This research was partly supported by the Strategic Information and Communication R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan.

References

1. Held, G.: Wireless mesh networks. Auerbach Publications, Boston (2005)
2. Gast, M.: 802.11 wireless networks: The definitive guide. O'Reilly Media, Inc., California (2005)
3. Challal, Y., Ouadjaout, A., Lasla, N., Bagaa, M., Hadjidj, A.: Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks. Journal of Network and Computer Applications 34(4), 1380–1397 (2011)
4. Wang, L., Seungho, J., Lee, T.: Redundant source routing for real-time services in ad hoc networks. In: IEEE International Conference on Mobile Ad hoc and Sensor Systems Conference 2005, pp. 80–87. IEEE Press, Washington, DC (2005)
5. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater acoustic sensor networks: research challenges. Ad Hoc Networks 3(3), 257–279 (2005)
6. Lun, D.S., Médard, M., Koetter, R., Effros, M.: On coding for reliable communication over packet networks. Physical Communication 1(1), 3–20 (2008)
7. Yang, Z., Li, M., Lou, W.: R-Code: Network coding-based reliable broadcast in wireless mesh networks. Ad Hoc Networks 9, 788–798 (2011)
8. Simulator, Q.N.: Scalable Network Technologies, `http://www.qualnet.com`
9. Peng, Y., Song, Q., Yu, Y., Wang, F.: Fault-tolerant routing mechanism based on network coding in wireless mesh networks. Journal of Network and Computer Applications 37, 259–272 (2014)
10. Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., Crowcroft, J.: XORs in the air: Practical wireless network coding. In: SIGCOMM 2006 Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 36(4), pp. 243–253. ACM, New York (2008)
11. Ikenaga, T., Tusbochi, K., Nobashi, D., Fukuda, Y.: Disjoint Path Routing for Multi-channel Multi-interface Wireless Mesh Network. International Journal of Computer Networks & Communications (IJCNC) 3(2), 165–178 (2011)

Community Roamer: A Social-Based Routing Algorithm in Opportunistic Mobile Networks

Tieying Zhu, Cheng Wang, and Dandan Liu

[1] School of Computer Science and Information Technology,
Northeast Normal University, China
zhuty@nenu.edu.cn

[2] School of Electrical Engineering and Computer Science,
University of Ottawa, Canada
cwan3@uottawa.ca

[3] School of Computer Science, Wuhan University, China
csliudd@gmail.com

Abstract. Opportunistic mobile networks enable mobile devices to exchange data by opportunistic contacts between devices. Since communication is highly impacted by human mobility, the social properties of human beings are exploited to improve the performance of data forwarding. In this paper, we first analyze the threshold configuration of community merging during the process of distributed community discovery. We then introduce a new social-based routing algorithm, called *Community Roamer*, which identifies those active nodes moving between different clustered communities to create an efficient inter-community forwarding path. Simulations on four real data sets show that the new algorithm has a much lower overhead than the existing algorithms and comparable message delivery ratio.

Keywords: Opportunistic social networks, social routing algorithm, community detection.

1 Introduction

Message delivery in opportunistic mobile networks depends on opportunistic contact between mobile devices. It enables mobile devices to exchange messages in the wireless range of each other in the absence of communication infrastructures. Communication in such networks is multi-hop and relies on the intermediate nodes to store and forward the message to the destination.

Opportunistic mobile ad hoc communication usually happens between the devices carried by human beings; the performance of such networks relies heavily on human mobility patterns and social behaviors. Research on mobility tracing data in different scenarios, including the campus life [1], conference [2], and virtual social networks [3], shows that the human mobility model is strongly correlated with social relationships among individuals. The contact rate, contact duration and inter-contact time in the tracing data reflect the contact frequency, longevity and regularity between people. On average, people tend to meet with family and friends frequently and regularly, with casual acquaintances less frequently and regularly, and with strangers randomly. Social relationships between people may vary less significantly than the topology of mobile networks. Those relatively stable characteristics alleviate the difficulties in effective data forwarding and dissemination due to unpredictable node mobility, rapid changing topology and lack of global network information. Based on those observations, some social-based characteristics, such as centrality, similarity and community, are employed to design opportunistic forwarding mechanisms [4], [5], [6], [7], [8], [9], [10]. In the literature, different social characteristics or their combinations are used as utility functions to rank which of the two nodes is more effective in forwarding or disseminating data.

In the social metrics used by routing algorithms, community is especially significant from a social perspective. Community structure reflects the clustering tendencies of human being and is relatively robust, since society itself offers a wide variety of possible group organizations: families, working and friendship circles and so on [11]. Community can be derived from the contact graph according to contacts between mobile devices in opportunistic mobile networks. In practice, however, it is challenging to discover a community locally and dynamically, since contacts between devices are opportunistic and time-varying.

In opportunistic mobile networks, it is important to consider how to discover the community structure and obtain accuracy similar to that of the centralised methods. At the same time, designing the forwarding metrics which reflect the data dissemination capability of nodes is also key to improving the performance of opportunistic routing. The paper explores these two problems from the point of view of theoretical analysis and experiments on real data sets. It uses conductance as the community evaluation metric to infer the threshold of community merging in the distributed community detection algorithm, which is proposed in [12]. Furthermore, a social-based routing algorithm, called Community Roamer, is proposed as an inter-community metric to guide routing decisions. Community Roamer routing uses the number of local communities that a node has been to as the utility function to evaluate the effect of a node in inter-community data forwarding.

The rest of the paper is organized as follows: Section 2 presents the related work, and Section 3 describes the distributed community detection method and the threshold configuration in the process of community discovery. The Community Roamer routing algorithm is introduced in Section 4, followed by the results of the experiment on data sets in Section 5, and the conclusion in Section 6.

2 Related Work

Many data forwarding algorithms in opportunistic mobile networks, or Delay Tolerant Networks (DTNs), have been proposed. Since there is no stable path between the mobile device pairs, all the current routing mechanisms use the store-carry-forward fashion. The routing decision is made locally by the message carrier, depending on partial knowledge of network topology and prediction of future contacts. In the evolution of relay selection approaches, the literature has fallen into three categories: flooding-based, encounter-based and social-based.

Flooding-based methods are simple and straightforward. The most popular algorithms are Epidemic routing [13] and Spray and Wait [14], which deliver the messages to all the encounters all the way or in the spray phase of routing. These algorithms are the most effective but have the highest cost. Encounter-based algorithms use the microcosmic viewpoint to evaluate the utility of relay candidates. They select relays directly according to contact information, such as the number of contacts, contact duration and inter-contact time. PROPHET routing [15] is one of the algorithms which use the past contact records to predict the message delivery probability to the destination.

Influenced by studies of social network analysis, social-based routing methods try to explore the various social properties from the contacts between mobile devices. These methods use the macroscopic viewpoint to discover relatively stable characteristics against volatile contacts to evaluate data forwarding capabilities. Label routing [4] takes advantage of the social communities of groups/affiliations as the community labels and selects the nodes which are in the same community as destination for relay. Bubble Rap routing [5] combines centrality and community to improve the performance of routing on the base of the distributed community detection. Since people have different roles and community structure may be overlapped [16], the recently developed Overlapping Community routing [7] explores overlapping community structures in mobile networks and forwards messages to those which share more common community labels with the destination. Friendship-Based routing [6] forwards the message to the node, which is in the same friendship community as the destination and has stronger friendship with the destination. The friendship community is a weighted temporal graph, in which the weight, defined as the social pressure metric, qualities the friendship between two specific node pairs and measures the average forwarding delay between them. Transient Community routing [10] explores the transient social contact patterns to improve forwarding performance. Besides the community-based routing, SimBet routing [9] relies on the combination of two social characteristics, the similarity between message carrier and the destination, and the ego betweenness centrality of the encounter nodes, to make routing decisions. Some literature explores other social properties related to routing in DTNs, such as PeopleRank routing [8]. It extends the idea of PageRank, which is used to evaluate the importance of web pages in webgraphs, and forwards messages to those nodes with higher PeopleRank value.

In the social-based routing algorithm, almost all of the literature tries to identify the highest ranking node in the contact graph as the message forwarder, according to contact rates, the number of contact neighbors or contact duration time. However, the recent work of Zyba et al. [17] emphasizes the roles of vagabonds in data dissemination when the vagabonds have large population and density under certain circumstances. Furthermore, Pietilainen et al. [18] show that devices with higher contact rate that spend most of the time within these social clusters do not impact data delivery performance as much as those nodes which have a medium number of contacts but spend less time in temporal communities. With these findings in mind, this paper proposes the Community Roamer routing algorithm which tries to find those nodes which are truly active and has strong forwarding capacity between communities.

At the same time, an efficiently distributed community detection mechanism is important for selecting a better relay node, since the community structure determines the scope for evaluating the forwarding utilities. The work in Overlapping Community routing [7] extends the overlapping clustering methods used in complex networks to a two-phase community detection framework for community detection and evolution in dynamic mobile networks. Although this work addresses the adding/deleting nodes and edges dynamically, it still needs global information during the basic overlapping community discovery. In Transient Community routing [10], broadcast is used to get community members to form the transient community at a certain moment. In [12], Hui et al. propose three distributed community detection methods, including SIMPLE, k-CLIQUE and MODULARITY, to identify the local community of each mobile node. In these three methods, MODULARITY exploits modularity evaluation of local communities proposed by Clauset [19], which extends the centralised global modularity [20] into the scope of distributed local community of each node. It is shown that MODULARITY has a computational complexity of $O(n^4)$ and has no significant improvement on performance compared with SIMPLE and k-CLIQUE method, while the complexity of these two methods is $O(n)$ and $O(n^2)$ at the worst case, respectively. Considering the computational and storage limit of mobile device, SIMPLE is competitive compared with the other two methods. However, it requires appropriate thresholds to determine the formation or combination of communities. The analysis in [21] demonstrates that SIMPLE with bad threshold configuration would lead to poor performance in Bubble Rap routing [5]. In the following section, the paper will analyze the threshold configuration and provide an analytical foundation for SIMPLE in distributed community detection.

3 Community Detection Threshold

Society of human beings is composed of clusters of family, friends or colleagues, etc. Communities are groups of people which probably share common properties and/or play similar roles within social networks [11]. Generally, nodes are strongly connected within a community and are more sparsely connected between communities. The conductance score is one of the simplest notions of community quality and can give the best performance in identifying ground-truth communities [22]. Conductance can be defined as the ratio between the number of edges inside the community and the number of edges leaving the community.

In a graph *G(V, E)*, S is a set of nodes, $S \in G$. The conductance of S is defined as $\varphi(S) = \frac{cs}{cs+2e}$, where $cs = |\{(u, v), u \in s, v \notin S\}|$, $e = |\{(u, v), u, v \in S\}|$ [23]. For two node sets, A and B, if $\varphi(A) < \varphi(B)$, node set A is more like a community than B since a good community should have better internal connections than those between communities.

In the process of community detection, each mobile node will keep a map of familiar set and local community. When the node v_0 meets the node v_i, each of them will check whether the other node can be included into the familiar set first. If the accumulation contact duration or number of contact exceeds a threshold, each node would join the other's familiar set. In this way, the node with only occasional contacts in temporal contact graph will be excluded from joining the other node's local community. Algorithm 1 shows the process of one node joining the other's local community in SIMPLE in Hui et al. work [12]. For example, for v_0, whether v_i can join in its local community LC_0 is determined by $\frac{|Fi \cap LC0|}{|Fi|}$, in which F_i is the familiar set of node v_i.

Algorithm 1. SIMPLE Community Detection

```
When node $v_i$ meeting node $v_0$
if $v_i \notin LC_0$ then
  if $|F_i \cap LC_0|/|F_i| > \gamma$ then
    $v_i \to LC_0$
  end if
end if
```

When considering the threshold of whether a node would join in the local community of another node, we compare the conductance before and after the merging. Fig. 1(a) and (b) show the community structures before and after merging. Supposing that the number of internal links in LC_0 is e_0, and the number of leaving edges is cs_0 except for those links with nodes ending in the set of $|F_i \cap LC_0|$, the conductance $\varphi(LC_0)$ before the node v_i joining in LC_0 is then shown as

$$\varphi(LC_0) = \frac{(cs_0 + |F_i \cap LC_0|)}{(cs_0 + |F_i \cap LC_0| + 2e_0)}.$$

After merging, the conductance is

$$\varphi'(LC_0) = \frac{(cs_0 + |F_i| - |F_i \cap LC_0|)}{(cs_0 + |F_i| - |F_i \cap LC_0| + 2(e_0 + |F_i \cap LC_0|))}$$

$$= \frac{(cs_0 + |F_i| - |F_i \cap LC_0|)}{(cs_0 + |F_i| + |F_i \cap LC_0) + 2e_0)}.$$

When $|F_i|$ is not too large and $|F_i| - |F_i \cap LC_0| \leq |F_i \cap LC_0|$, i.e. $|F_i \cap LC_0|/|F_i| \geq 1/2$, then $\varphi'(LC_0) < \varphi(LC_0)$. This means that when the proportion of the intersection of the local community of v_0 and the neighbor set of v_i in the neighbor of node v_i is equal to 1/2 or more, the merge would make the internal structure of v_0's local community more tight than before. Therefore, from the view of community structure, a untight lower bound of one node joining the other's local community is 1/2 when $|F_i|$ is not too large.

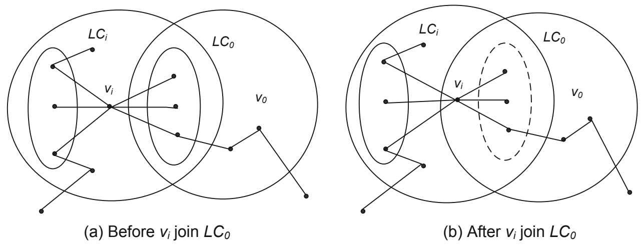


Fig. 1. Community structures before and after v_i joining LC_0

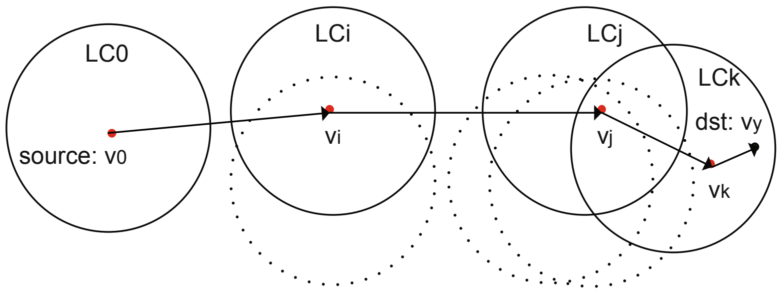


Fig. 2. Community Roamer routing. Each solid circle denotes the local community of the node which is marked in red and the dashed circle denotes the local community of the other nodes which the red node belongs to. The source of the message is v_0, and the destination is v_y. The inter-community forwarding path is $v_0 \to v_i \to v_j \to v_k$, and the intra-community path is $v_k \to v_y$.

4 Community Roamer Routing

In opportunistic networks, data forwarding relies on opportunistic contacts. According to community properties shown on the mobile devices, the community-based data forwarding would be carried out in two phases: inter-community forwarding and intra-community forwarding. Inter-community forwarding occurs when the local community of message carrier does not contain the destination. In this case, the message should be delivered as soon as possible along the path to the community containing the destination. Therefore, the forwarding utilities in inter-community forwarding typically rely on global rankings, such as the global degree centrality in [5], the number of overlapping communities with the destination in [7] and betweenness centrality used in [9]. Intra-community forwarding means the data forwarding when both the local communities of message carrier and the encounter node contain the destination. In this case, messages are only forwarded inside the community in order to deliver to the destination as fast as possible. Intra-community forwarding typically relies on local ranking, such as the higher number of encounters in the local community [5]. The paper proposes Community Roamer routing algorithm which focuses on inter-community forwarding and uses a new routing metric that reflects the data dissemination

Algorithm 2. Community Roamer routing

```
When message $m's$ carrier $v_0$ meets node $v_i$
if $v_i$ merged into $LC_0$ then
    number of communities $v_i$ traveled ++;
end if
if $v_0$ merged into $LC_i$ then
    number of communities $v_0$ traveled ++;
end if
if $dst(m) == v_i$ then
    deliver message $m$ to $v_i$
else if $dest(m) \in LC_0 \&\& dest(m) \in LC_i$ then
    if $v_i$ has large local centrality than $v_0$ then
        select $v_i$ as relay;
    end if
else if $dst(m) \notin LC_0 || dst(m) \in LC_i$ then
    select $v_i$ as relay;
else if $dst(m) \notin LC_0 || dst(m) \notin LC_i$ then
    if $v_i$ traveled more communities than $v_0$ then
        select $v_i$ as relay;
    end if
end if
```

capability between clusters. The intuition behind the idea is that the node which has been to a large number of communities has larger inter-community forwarding capabilities, since they are more active and roam between clustered communities, like a postman traveling between different communities. In the community detection process, a node can join in the local community of the encounter node if the threshold is met. Therefore, those nodes with more encounters or more acquaintances will have more chances to join in local communities of the other nodes. When quantifying the number of local communities to which a node have been, a larger number means larger capabilities of data forwarding between communities. Fig. 2 demonstrates of details Community Roamer routing. When two nodes meet each other, the node having the higher number of traveled communities will be selected as the relay until the message enters the community containing the destination and is delivered to the destination. The details of routing method are described as Algorithm 2.

5 Experiments

5.1 Data Sets and Parameter Configuration

In this paper, we use four experimental data sets: MIT Reality, Infocom06, Infocom05 and Cambridge to evaluate the effectiveness of the Community Roamer routing algorithm. These data sets are modified as encounter traces for ONE simulator [24] [25]. They include two campus environments: MIT Reality and Cambridge, and two conference environments: Infocom05 and Infocom06. MIT

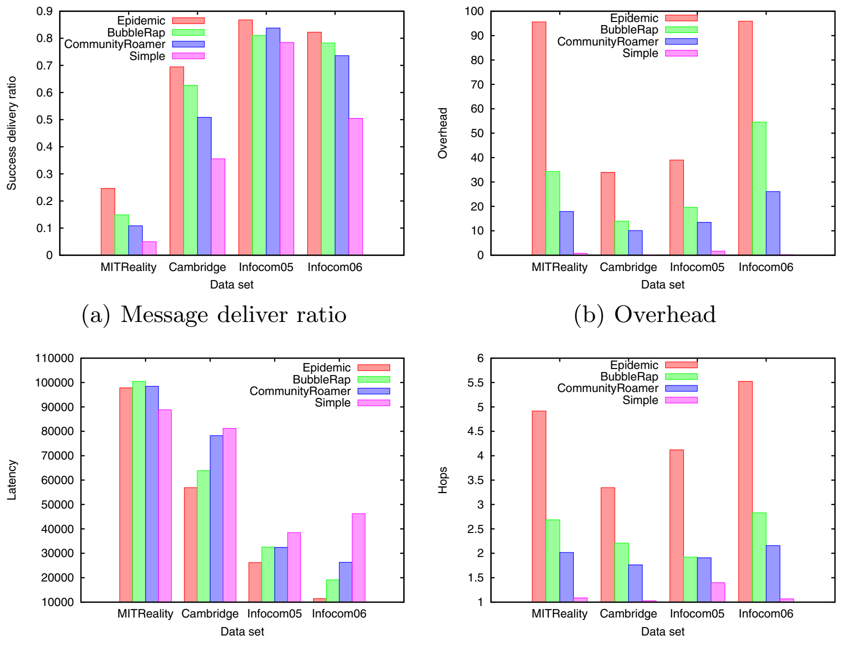
Reality has 97 participants within a period of nearly 9 months. The Cambridge data set has 36 experimental devices in 11 days. Infocom05 and infocom06 record the contacts during the INFOCOM conference in 2005 and 2006, respectively. The former involves 41 mobile devices and the latter includes 20 static devices and 78 mobile devices. In simulation, the threshold for a node joining the local community of the other is set at 1/2. The message is created randomly every 300-360 seconds. The length of each message is 0.5KB, and the memory of each node is 500MB. The message time-to-live (TTL) is set at 1 hour. Since the contact is sparse in MIT Reality data set, the success-delivery ratio/succesful is relatively lower than that of the other three data sets.

5.2 Results

In this section, the Community Roamer routing algorithm is compared to epidemic routing [13], Bubble Rap routing (including both the inter-community delivery and intra-community delivery) [5] and Bubble Rap with only intra-community delivery, called Simple routing, which means the messages are forwarded only when the local communities of message holder and encounter both contain the destination. For intra-community forwarding, the local degree centrality is used as the ranking function. For inter-community forwarding, the global degree centrality is used in Bubble Rap. Fig. 3 shows the results of success delivery ratio, overhead, latency and the average number of hops of the delivered messages on four data sets, respectively.

The message delivery ratio evaluates the percentage of the number of messages delivered within the total number of messages created. Epidemic routing has the best delivery performance since it explores all the possible paths to the destination. The Simple routing should have the lowest performance since it explores only intra-community transmission. The delivery performance of Bubble Rap and Community Roamer should be in between Epidemic routing and Simple routing. As shown in Fig. 3(a), for Infocom05 data set, Community Roamer is the best except for epidemic routing, and has a successful delivery ratio of 83.8%, while the performance of Bubble Rap is 81.1%. For the Infocom06 data set, the performance of the Community Roamer is 73.6%, 4% lower than that of Bubble Rap. For MIT Reality, the performance of Bubble Rap and Community Roamer are 14.9% and 10.9%, respectively. They are all much higher than that of Simple routing, which is 4.9%. The results mean that most messages are delivered in inter-communities forwarding for MIT Reality data. Similar results are found for Cambridge.

Fig. 3(b) shows the overhead of each methods, defined as the ratio of all the useless relay in the relay leading to successful delivery. The overhead measures the cost of each delivered message. The overhead of Epidemic routing is the largest since it is based on flooding. Simple routing has the lowest overhead since it only relays messages inside the community. In Simple routing, almost all relays lead to the successful delivery on Infocom06 and Cambridge data. In contrast to Bubble Rap, Community Roamer has much lower overhead on all the four data sets. Especially for the Infocom05 data set, the overhead of the



(a) Message deliver ratio (b) Overhead

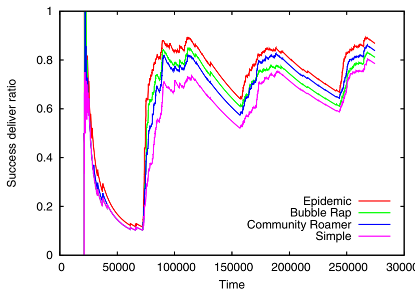
(c) Latency (d) Hops

Fig. 3. Routing performance on four data sets in terms of (a) message delivery ratio, (b) overhead, (c) latency and (d) hops

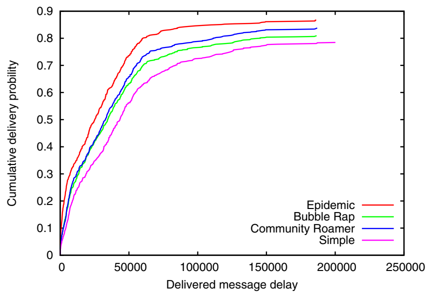
Community Roamer is 13.46, while the overhead of Bubble Rap is 19.61. At the same time, the success delivery ratio of the Community Roamer is 83.8%, while that of Bubble Rap is 81.1%. This means that Community Roamer routing has better forwarding performance with fewer relays than the Bubble Rap for Infocom05 data set. On Infocom06, the overhead of the community roamer is only 47.8% of that of Bubble Rap, while the successful delivery ratio is just 4% lower than that of Bubble Rap.

Message delivery latency measures the delay from the time/when a message is created to the time a message is delivered. The average number of hops of the delivered messages measures the length of delivery path. Fig. 3(c) and Fig. 3(d) show the latency and the number of message delivery hops, respectively. Except for Infocom05, the latency in Community Roamer is slightly larger than Bubble Rap, while the hops taken by messages in community roamer is lower than Bubble Rap in all four data sets.

For the Infocom05 data set, Fig. 4(a) and Fig. 4(b) further illustrate the changing of message delivery ratio with simulation time and the cumulative delivery probability sorted by delivered messages' delay. It is shown that the performance of Community Roamer outperforms Bubble Rap over time for Infocom05 data.



(a) Message delivery ratio



(b) Cumulative delivery probability

Fig. 4. Routing performance for Infocom05 data

In order to analyze the effectiveness of the Community Roamer routing algorithm further, we compare the global degree centrality of each node which is used in Bubble Rap routing, and the number of community to which each node traveled, which is used in Community Roamer routing for Infocom05 data. Fig. 5 illustrates the value of the number of each node traveled and the global degree centrality at the end of the temporal contact graph, respectively. For each node from 0 to 40, the global degree centrality and the number of communities to which each node traveled are not positively correlated. For example, node 31 has been to 15 local communities, the largest number, while its global degree centrality is 39, and many other nodes' global degree centrality is 40. Node 30 has the smallest number of acquaintance, 21, but has traveled to 3 communities. Further, it is shown that many nodes has the largest degree centrality, 40, which is nearly equal to the size of the Infocom05 data set. It means that the cumulated contact graph expands to the entire network over time when calculating degree centrality. On the other hand, the number of the local communities to which a node has traveled is far less than the size of the data set due to the use of

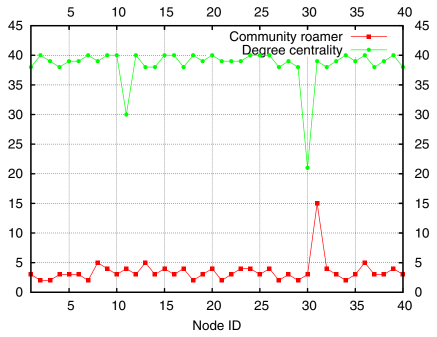


Fig. 5. Comparison of the number of communities to which each node traveled and the degree centrality of each node for Infocomm05 data

threshold. According to the analysis of the performance mentioned in Fig. 3 and 4, the performance of Community Roamer outperforms the Bubble Rap routing for Infocom05 data. In this sense, the metric of the number of communities to which a node has traveled accurately qualifies the inter-community forwarding capability of nodes.

6 Conclusion

This paper analyzes the threshold configuration in the process of local community formation using conductance. At the same time, a social-based routing algorithm, Community Roamer, is proposed and evaluated. Community Roamer routing uses the number of local communities to which a node have traveled to evaluate the data dissemination capability of a node for inter-community forwarding. The experiments on four data sets show that the community roamer has much lower overhead than Bubble Rap but comparable delivery performance. Especially for Infocom05, it outperforms Bubble Rap in all of the terms of success delivery ratio, overhead, latency and hops of delivered messages.

The Community Roamer is designed for inter-community message delivery. Its efficiency depends on the accuracy of detected community structure. The current evaluation is done on the cumulative contact graph and SIMPLE method following the work of Hui et al. Future work will consider designing a new effective distributed community detection for dynamic networks to reflect the evolution of community structures and evaluate community roamer routing algorithm in temporal communities.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (No. 61103216), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20110141120038), the Special Fund for Fast Sharing of Science Paper in Net Era by CSTD (FSSP 2013), and the Twelfth Five-Year Plan of Education Department of Jilin Province.

References

1. Eagle, N., Pentland, A.: Reality mining: Sensing complex social systems. Personal and Ubiquitous Computing 10(4), 255–268 (2006)
2. Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., Scott, J.: Impact of human mobility on the design of opportunistic forwarding algorithms. In: Proceedings of the 29th Conference on Information Communications (INFOCOM 2006), pp. 1–13 (2006)
3. Varvello, M., Voelker, G.M.: Second life: A social network of humans and bots. In: Proceedings of the 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2010), pp. 9–14 (2010)
4. Hui, P., Crowcroft, J.: How small labels create big improvements. In: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW 2007), pp. 65–70 (2007)

5. Hui, P., Crowcroft, J., Yoneki, E.: Bubble rap: Social-based forwarding in delay tolerant networks. IEEE Transactions on Mobile Computing 10(11), 1576–1589 (2010)
6. Bulut, E., Szymanski, B.K.: Exploiting friendship relations for efficient routing in mobile social networks. IEEE Transactions on Parallel and Distributed Systems 23(12), 2254–2265 (2012)
7. Nguyen, N.P., Dinh, T.N., Tokala, S., Thai, M.T.: Overlapping communities in dynamic networks: Their detection and mobile applications. In: Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MobiCom 2011), pp. 85–96 (2011)
8. Mtibaa, A., May, M., Diot, C., Ammar, M.: Peoplerank: Social opportunistic forwarding. In: Proceedings of the 29th Conference on Information Communications (INFOCOM 2010), pp. 111–115 (2010)
9. Daly, E.M., Haahr, M.: Social network analysis for information flow in disconnected delay-tolerant manets. IEEE Transactions on Mobile Computing 8(5), 606–621 (2009)
10. Gao, W., Cao, G., Porta, T.L., Han, J.: On exploiting transient social contact patterns for data forwarding in delay-tolerant networks. IEEE Transactions on Mobile Computing 12(1), 151–165 (2013)
11. Fortunato, S.: Community detection in graphs. Physics Reports 486, 75–174 (2010)
12. Hui, P., Yoneki, E., Chan, S.Y., Crowcroft, J.: Distributed community detection in delay tolerant networks. In: Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch 2007), pp. 1–8 (2007)
13. Vahdat, A., Becker, D.: Epidemic Routing for Partially-Connected Ad Hoc Networks. Duke University (2000)
14. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: Proceedings of the, ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN 2005), pp. 252–259 (2005)
15. Lindgren, A., Doria, A., Schelén, O.: Probabilistic routing in intermittently connected networks. SIGMOBILE Mob. Comput. Commun. Rev. 7(3), 19–20 (2003)
16. Palla, G., Imre Derenyi, I.F., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814–818 (2005)
17. Zyba, G., Voelker, G.M., Ioannidis, S., Diot, C.: Dissemination in opportunistic mobile ad-hoc networks: The power of the crowd. In: Proceedings of the 29th Conference on Information Communications (INFOCOM 2011), pp. 1179–1187 (2011)
18. Pietiläinen, A.K., Diot, C.: Dissemination in opportunistic social networks: The role of temporal communities. In: Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2012), pp. 165–174 (2012)
19. Clauset, A.: Finding local community structure in networks. Physical Review E (2005)
20. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69 (2004)
21. Orlinski, M., Filer, N.: The rise and fall of spatio-temporal clusters in mobile ad hoc networks. Ad Hoc Networks 11(5), 1641–1654 (2013)
22. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics (MDS 2012), pp. 1–8 (2012)

23. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transcations of Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
24. Keränen, A., Ott, J., Kärkkäinen, T.: The one simulator for dtn protocol evaluation. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools 2009), pp. 1–10 (2009)
25. Orlinski, M.: `http://www.shigs.co.uk/connection-traces-for-the-one-simulator/`

A Self-adaptive Reliable Packet Transmission Scheme for Wireless Mesh Networks

Wenze Shi[1,2], Takeshi Ikenaga[2], Daiki Nobayashi[2], Xinchun Yin[1], and Hui Xu[1]

[1] School of Information Engineering, Yangzhou University, Yangzhou, China
swz4123@qq.com
[2] Graduate School of Engineering, Kyushu Institute of Technology, Fukuoka, Japan
{ike,nova}@ecs.kyutech.ac.jp

Abstract. Packets transmission in Wireless Mesh Networks (WMNs) is not that reliable due to the inherited weak points of wireless channels. In this paper we proposed a new packets transmission scheme which is called – Self-adaptive Reliable Transmission scheme with Redundancy and Multipath (S-RTRM). S-RTRM is improved on the foundation of RTRM which uses XOR operation as its encoding method and rank the redundancy strategy in 5 standard scales. Different from RTRM, S-RTRM uses new path metric and can monitor each path's performance in real-time. Then it can choose a proper redundancy strategy dynamicly for each path. Any path will undertake its own redundant packets. Compared with RTRM, S-RTRM is effectual in a much larger data rate area and doing its best on weighing the reliability against good put.

Keywords: Wireless mesh networks, redundant packets, multipath routing, reliable transmission, self-adapting packet transmission scheme.

1 Introduction

Wireless Mesh Network (WMNs) is a kind of distributed, self-organizing, multi-hop wireless network which is made up of radio nodes organized in a mesh topology [1]. These features bring many characteristics like easy and low cost deployment, larger coverage, powerful self-healing ability, etc. As a new type of broadband wireless access network, WMNs aims at providing networks in anytime anywhere. All these advantages attract a lot of attentions on it. However the nature weaknesses of wireless channel such as high noise, fast fading and easy to suffer interference make the packets transmission on WMNs not that reliable [9]. For example, when there is interference, Signal-to-Noise Rate (SNR) will decreases at the receiver and the packet loss rate will be highly increased. In this case, how to make the packets transmission more reliable on WMNs becomes a research hotspot in recent years [5].

Since Reliable packet Transmission with Multipath and Redundancy (RTRM) [12] is sensitive to data rate, researches were done focus on packets loss and the basic reasons of packets loss were found. According to these, we can prevent packets loss in Network layer by control the original data stream to each path. Then we choose a proper redundancy strategy for each to recover lost packets in Mac layer. Based on these theories a Self-adaptive packets transmission scheme S-RTRM is proposed.

The rest of this paper is organized as follows. Section 2 introduces some related works. Section 3 shows the research process on packet loss. In Section 4, we give the implementation of S-RTRM. Section 5 shows the performance evaluations of the proposed scheme. Finally, conclusion and future works are given in Section 6.

2 Related Works

Network fault-tolerance strategy in software level includes network coding [6], data fusion, multipath routing [7], adding redundant packets, etc.

Multipath Source Routing (MSR) was proposed by Lei Wang [3][4]. MSR combines multipath routing and adding redundant packets together to ensure the reliable transmission in Mobile Ad hoc NETwork (MANET). However it only uses two paths to transmit packets and use duplicate packets as redundant packets. The increment of reliability by using two paths is restricted and using duplicate packets will increase 100% of network load. What's more they send original packets and duplicate packets in different paths. But they do not denote which way is more suitable for redundant packets. And they use RTT as multipath metric [8]. RTT cannot estimate the packet loss rate of a path so that they may not able to choose a proper path to use.

Reliable Transmission scheme with Redundancy and Multipath (RTRM) also combines multipath routing [11] and adding redundant packets together. It uses XOR operation as its encoding method and rank the redundancy strategy in 5 standard scales. Different redundancy strategy has different redundant rate. Usually we think that higher redundant rate leads to higher reliability. But RTRM proves that this rule is just suitable in low data rate scenarios. In high data rate scenarios, light redundancy strategy maybe lead to better performance. It means we only need to add 20% extra packets which will get higher reliability than that adds duplicates into network traffic. What's more RTRM treats redundant packets the same as original packets and distributes them to each path by round robin. It also proved that round robin is the best way to transmit redundant packets.

Even RTRM get big progress in balancing the reliability and good put, it still has some problems. We found RTRM is a data rate sensitive scheme. Since the rule that higher redundant rate leads to higher reliability is only useful in low data rate scenarios, it will be difficult for RTRM to choose a redundancy strategy in all cases. The reason why heavy redundancy strategy causes worse performance in reliability is unknown. This paper will unravel the basic reason of packet loss and find out why RTRM is sensitive to data rate. After solving these problems a self-adaptive reliable packets transmission scheme S-RTRM was proposed. Simulation proves that S-RTRM can provide reliable transmission in a much larger data rate area compared with RTRM.

3 Research on Packet Loss

In order to simulate in more complex scenario, we decide to control the SNR of a wireless link by changing Transmission Power (Tx-Power) at the sender. If so, we need to get the relation between Packet Delivery Ratio (PDR) and Tx-Power first. Since we use Tx-Power to control the change of SNR, the distance between two nodes does not need to change. That distance is fixed at 5 meters. Fig.1 shows the relation between PDR and Tx-Power of different bandwidths in IEEE 802.11a standard. The simulation scenario is a two nodes wireless link. And CBR at the sender is 1Mbps. We believe that we can set any wireless link's PDR freely depends on Fig.1 However, in our latter simulations their PDR is not what we expected. Then we did the same simulation but on different CBR. Fig.2 is the relation between PDR and Tx-Power when CBR is 20Mbps. This figure shows most of the curves are totally different from Fig.1. This phenomenon also proves that PDR is sensitive to data rate.

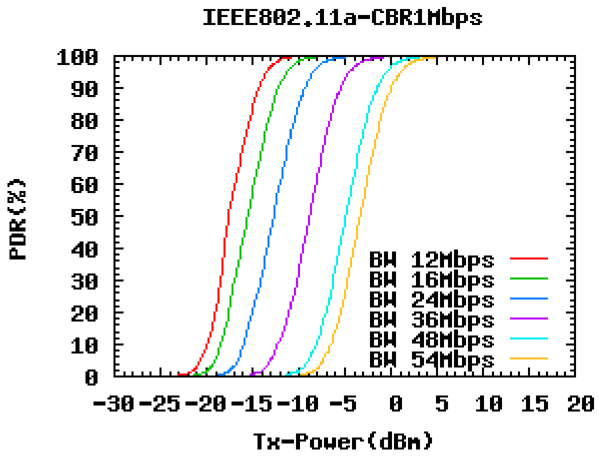


Fig. 1. PDR to Tx-Power of IEEE 802.11a when CBR is 1Mbps

In order to get clear about the relation between PDR and data rate in a particular bandwidth, we did more simulations in a two nodes wireless link by changing its Bandwidth and CBR. Fig.3 shows the simulation results when bandwidth is 12Mbps. We can see from this figures that when Tx-Power is fixed, generally higher data rate have lower PDR. Based on previous research, we infer that the decrease of Tx-Power causes more packet loss, and the MAC layer of IEEE 802.11 standards has a unicast retransmission mechanism. Retransmission needs more time, and if the data rate is very high, the sender will not be able to send all this packets, some packets would be lost. In order to prove our inference, we did more simulation and not only focus on PDR but also paid attention to some MAC layer and Physic layer statistics. These statistics are shown in Table 1. We found two basic reasons of packet loss as Fig.4 shows.

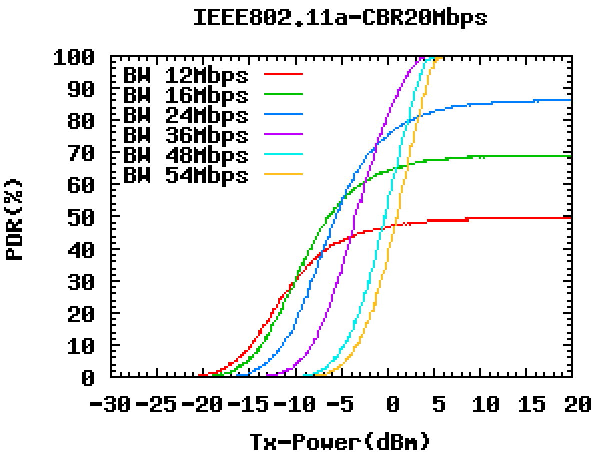


Fig. 2. PDR to Tx-Power of different Bandwidth when CBR is 20Mbps

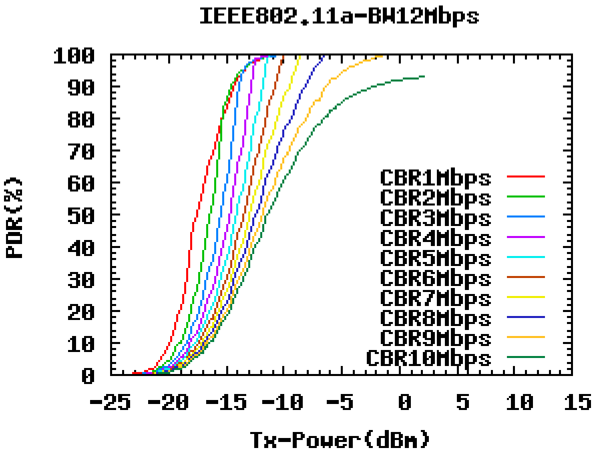


Fig. 3. PDR to Tx-Power of different CBR when Bandwidth is 12Mbps

Table 1. Some important Mac and Physic Layer simulation statistics of QualNet

Layer	Name of Statistics
Transport	Packets from Application Layer = 10000
Network	Total Packets Queued = 2873
MAC	Unicast packets sent to channel = 2864
MAC	Packet retransmissions due to ACK timeout = 1806
MAC	Packet drops due to retransmission limit = 9
Physical	Signals transmitted = 4670

Reason1: When Tx-power decreases the SNR decreases either, and it makes the receiver's MAC layer more difficult to recover the frames. This causes some packets lost and we call it Frame Error Rate (FER) of a wireless channel.

Reason2: Between the Network layer and Data link layer there is a queue. When FER increases the MAC layer have to retransmit frames incessantly. That causes the ability of a sender to send frames becomes weaker. And then the queue becomes full and causes overflow. Here we call it Network layer Packet Loss Rate (NPLR).

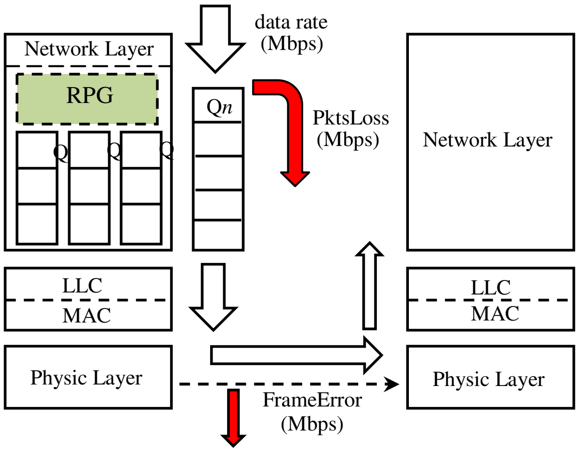


Fig. 4. Two basic reasons cause packet loss

At the same time, because of the unicast retransmission mechanism, there are two kinds of FER. One is Physic layer FER (PhyFER) and the other is MAC layer FER (MacFER). Fig.5 shows the PhyFER and Fig.6 shows the MacFER. Because our scheme is located between Network layer and Data Link layer, here we only consider MacFER.

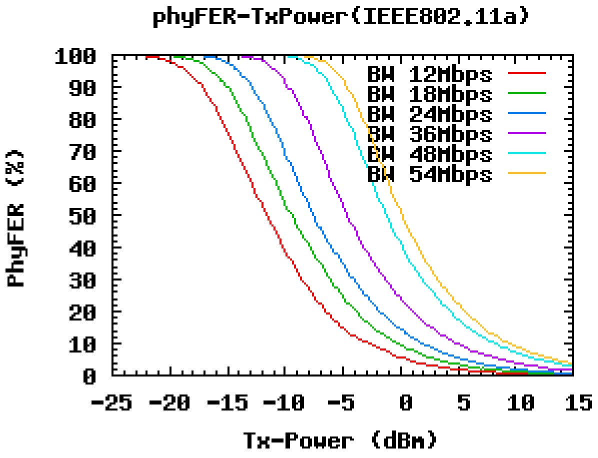


Fig. 5. PhyFER to Tx-Power of different Bandwidths of IEEE 802.11a

Since we find out the basic reasons of packet loss, we can get the formula of PDR as shown in formula (1):

$$PDR=(1-NPLR)(1-MacFER) \quad (1)$$

To increase PDR, we only need to decrease NPLR and MacFER. From reason 2 we know that, when Tx-Power is fixed, the ability of a queue to pass packet is fixed too. Since this ability looks like a window size, we call it Queue Window Size (QWS). Then we find out the relation between QWS and Tx-Power as Fig.7 shows.

Assume that the Tx-Power is fixed, if we can control the data rate of the sender no larger than its QWS, then no packet will loss before entering the queue. In this case, the NPLR can be controlled to 0%. Now the PLR can be written as formula (2):

$$\text{PLR=MacFER} \quad (2)$$

Table 2. Details of 5 redundancy strategies

strategy	*n*	*Redundancy rate* (%)	*Delivery threshold* (%)
1	1	100	90
2	2	50	91.5
3	3	33	93
4	5	20	95
5	10	10	96.5

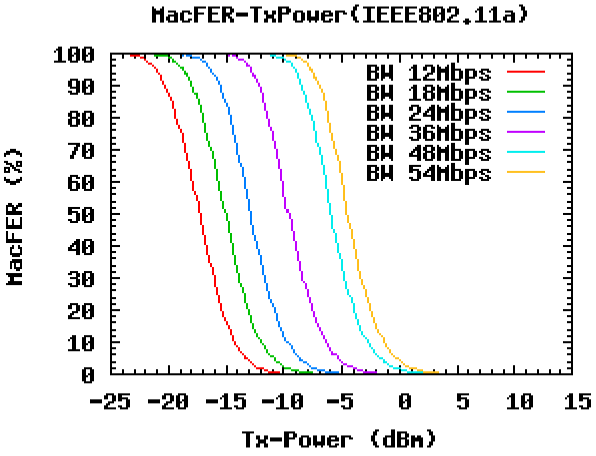


Fig. 6. MacFER to Tx-Power of different Bandwidths of IEEE 802.11a

In addition, we combine Fig.6 with Fig.7 and find that the horizontal axes of these two figures are the same. That means there must be a mapping relation between QWS and MacFER. If we know the value of QWS we can get the value of MacFER. From formula (2) we know MacFER equals to PLR. Table 2 shows some details of these 5 redundancy strategies. It includes:

n: the number of original packets in a group

Redundancy rate: equals to $\frac{1}{n}$.

Delivery threshold: it means if the PDR is larger than this threshold, we use this strategy can increase the PDR up to 99%.

Table 2 is referenced from [12]. Depending on Table 2, if the PLR is no bigger than 10%, we can choose a proper redundancy strategy to make it smaller than 1%.

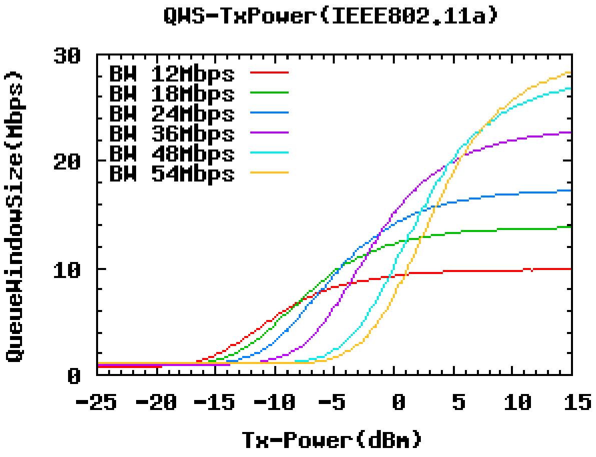


Fig. 7. QWS to Tx-Power of different Bandwidths of IEEE 802.11a

4 Proposed Scheme

4.1 Network Model

Fig.8 shows the network model of S-RTRM. Our attention is still between Network layer and Data link layer. Different from RTRM, S-RTRM adds a Redundant Packet Generator Manager (RPGM) firstly, and under the RPGM there are several RPGs. One path has one RPG. The green packets represent redundant packets.

Functions of RPGM:

- Choose a proper redundancy strategy. Different path use different redundancy strategy.
- Counting the capacity of a path depends on its QWS.
- Distribute network traffic. According to the capacity rate among all paths, RPGM distribute a different number of original packets to each path.

Functions of RPG:

- Generate redundant packets according to the redundancy strategy chosen by RPGM.

Functions of RPF:

- Recover lost original packets.
- Filter out redundant packets.

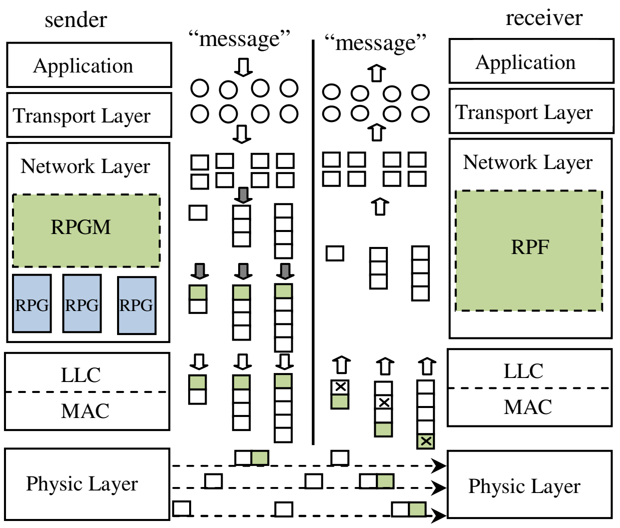


Fig. 8. Network model of S-RTRM

4.2 Process of S-RTRM

Step 1: Use Source Routing Protocol to discover available paths, and choose at most 3 node disjoint paths. Firstly we choose the path with minimum MacFER. If their MacFER are the same then we choose the path with maximum QWS.

Step 2: RPGM will choose a proper redundancy strategy for each path according to QWS and MacFER. Then *n*, the number of original packets in a group is decided. At the same time, RPGM will count the maximum original packet data rate for each path, and we define it as CBR_i for Path *i*. It follows the formula (3)

$$CBR_i = \frac{n}{n+1} QWS \qquad (1 \leq i \leq numOfPaths) \tag{3}$$

Step 3: All packets go through RPGM can get a FlowID. And RPGM will distribute a PathID to each packet by running traffic distribution algorithm.

Step 4: When these packets go through RPG, they will be divided by FlowID and get a SeqID. Then they will be classified into groups. Each group has *n* packets, and each packet has a same GroupID.

Step 5: RPG will have all packets in a same group done XOR operation and get a redundant packet. Redundant packet has the same GroupID and a redundancy flag .

Step 6: In RPF, there is a two-dimensional structure array, and the subscript is FlowID and GroupID. Each structure includes 4 elements:

- The number of group packets *n*.
- A *counter* to calculate how many packets in the same group have been received.

- A *recover* which has the same size as original packet. Its initialization value is 0.
- A *flag* which indicate if a group has received a redundant packet. If that group has received a redundant packet, *flag* sets to1, otherwise *flag* sets to 0.

When any packet arrives, it can find a particular structure for its group by checking the subscript of PathID and GroupID. Then RPF will make the arrived packet and *recover* do XOR operation and record the value in the *recover*. After that if the packet is a redundant packet, RPF will filter it out. Otherwise, RPF will submit it to upper layer.

Step7: the transmission result of a group has the following 3 possibility:

- If the value of *counter* equals to n+1, that means all packets have been delivered.
- If the value of *counter* equals to n, we check the value of *flag*. If *flag* equals to 1, that means 1 original packet is lost. Then we submit recover to upper layer as the lost packet.
- If the value of *counter* is littler than n, that means in a group more than 1 original packets were lost. In this case we will do nothing.

4.3 Traffic Distribution Algorithm

Because S-RTRM distribute original packets depends on the condition of each path, so different path may undertake different amount of network traffic. Therefor how to distribute the traffic has significant meanings. For example there are 3 paths and their capacity rate is $CBR_1\colon CBR_2\colon CBR_3 = 1\colon 2\colon 3$. If their distribution algorithm doesn't follow the rate in time which just like Fig.8 shows. Since the real time data rate of each path dose not decrease, it will cause overflow at the queue and more packets will loss.

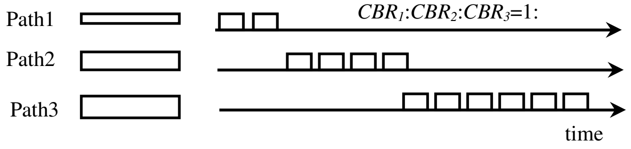


Fig. 9. Packets transmission sequence of a sample distribution algorithm

To solve this problem we proposed a new distribution algorithm based on the accumulation of path step length. We take the condition in Fig.8 as an example. In that example $CBR_1\colon CBR_2\colon CBR_3 = 1\colon 2\colon 3$, so we define sum as formula (4)

$$Sum = \sum CBR_i \quad (0 < i < number\ of\ paths) \tag{4}$$

Process of Traffic distribution

Step 1: Compute the $steplength$ of Path$i$ as formula (5):

$$steplength_i = \frac{sum}{CBR_i} \tag{5}$$

Step 2: Each path has a *counter* for accumulating its $steplength$. Every time Path i send a packet, the value of $counter_i$ will plus $steplength_i$. If the value of $counter_i$ equals to *sum*, $counter_i$ will be set to 0.

Step 3: The path which has the minimum value of $counter_i$, has the highest priority to send packet.

Step 4: Otherwise the path which has the minimum of $steplength_i$ will has the highest priority to send packet.

Step 5: If the $steplength$ of all paths are equal, the path with the minimum PathID will sent packet first.

Fig.9 shows the packets transmission sequence of our traffic algorithm. The real time data rate of each path absolutely follows their capacity rate.

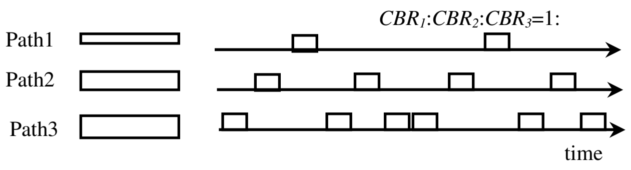


Fig. 10. Packets transmission sequence of our distribution algorithm

5 Performance Evaluations

5.1 Scenario Topology

All of our simulations were held on QualNet 4.5 simulator [10].We assume to have 3 nodes disjoint paths from a source node to destination node. The distance between each neighbor node is 5 meters. We set their Bandwidth and MacFER as Fig.10 shows.

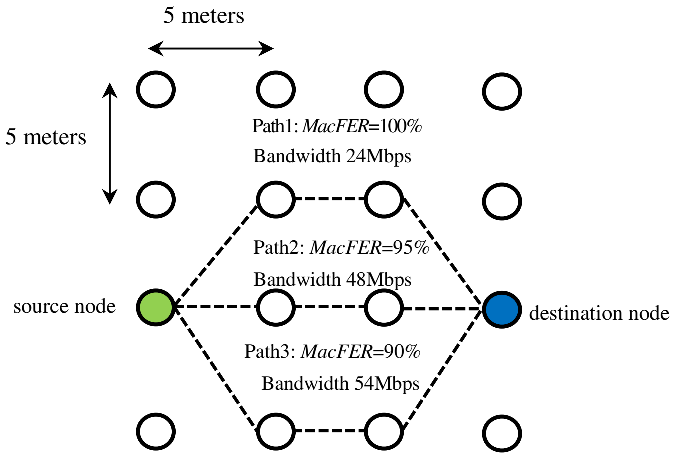


Fig. 11. Topology of simulation scenario

We test the performance of RTRM and S-RTRM respectively when CBR changes from 1Mbps till 30Mbps. Fig.11 compares the PDR of RTRM and S-RTRM when CBR changes.

5.2 Simulation Results and Analysis

Simulation results are seen in Fig.11, before CBR is 10Mbps, RTRM can provide reliable transmission. However, when CBR keeps increasing RTRM's PDR decreases fast. This phenomenon is obvious due to previous research. When CBR is smaller than 10Mbps, even RTRM add some redundant packets into each path, the total data rate of each path dose not exceeds its QWS. However, when CBR keeps increasing, bad path losses a lot of packets since its data rate exceeds its QWS, while good path has poor efficiency because the amount of redundant packets is decided by bad path. Many of these redundant packets are unnecessary for good path. Therefore, when CBR is larger than 10Mbps, its PDR decrease fast.

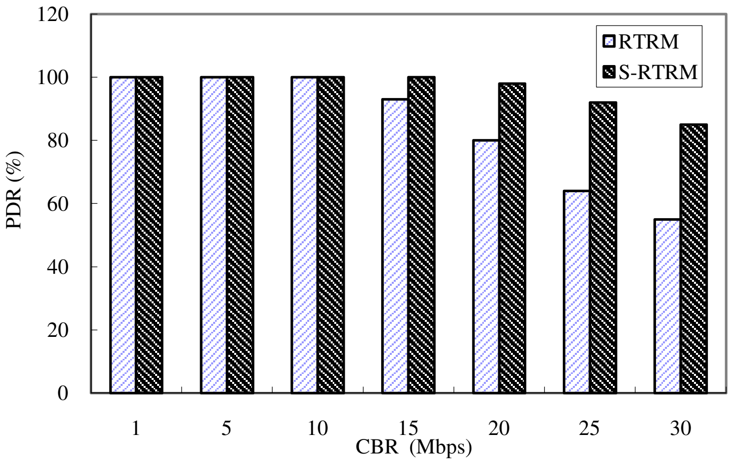


Fig. 12. Comparison of RTRM and S-RTRM when CBR changes from 1Mbps to 30Mbps

For S-RTRM, when CBR is 20Mbps the PDR can get to 99%.That is due to the control of network traffic to each path. S-RTRM controls the original data rate to each path, this can keep the data rate always below its QWS.S-RTRM also chooses appropriate redundancy strategy for each path to make the transmission reliable. After 20Mbps the PDR starts to decrease. It is because there is no perfect transmission scheme which is impactful in any scenario for any data rate. S-RTRM extends the effectual data rate area from 10Mbps to 20Mbps compared with RTRM.

From the comparison we draw the conclusion that S-RTRM not only can improve the reliability of transmission in wireless mesh network but also be effectual in a larger data rate area.

6 Conclusion and Future Works

In this paper, a new self-adaptive reliable transmission scheme for wireless mesh network, S-RTRM is presented. S-RTRM is evolve from RTRM but much better than RTRM. During the process of design S-RTRM, we found out the basic reason of packet loss. Based on these findings our S-RTRM can adapt to each path's own condition and make a best transmission scheme respectively. S-RTRM does its best to balance redundancy and good put. The simulation results also prove that our S-RTRM can provide reliable transmission for wireless mesh network in a much larger data rate area.

Even though we find out the basic reason of packet loss and can control the network traffic effectively, we have other findings but don't know the reason. From Fig.3 we know that at a fixed Tx-Power value, when the CBR increase the PDR decrease. It is because at this point the capacity of the wireless link is fixed. Higher CBR can cause more packets loss at the queue. However this rule sometimes doesn't work especially when the CBR is very small thus cannot cause overflow and there is MacFER but not very large. Under this circumstance higher CBR usually get higher PDR. We can see this from Fig.3 when Tx-Power equals to -16 dBm, 2Mbps CBR has higher PDR than 1Mbps CBR. And this phenomenon is more serious in large Bandwidth like 54Mbps. This causes the PDR of multipath lower than single path when the original data rate very small. So our Future work will focus on finding the reason of this phenomenon.

Acknowledgement. This research was partly supported by the Strategic Information and Communication R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan.

References

1. Held, G.: Wireless mesh networks. Auerbach Publications, Boston (2005)
2. Gast, M.: 802.11 wireless networks: The definitive guide. O'Reilly Media, Inc., California (2005)
3. Wang, L., Zhang, L., Shu, Y., Shu, Y., Dong, M.: Multipath source routing in wireless ad hoc networks. In: Electrical and Computer Engineering 2000 Canadian Conference, vol. 1, pp. 479–483. IEEE Press, Canadian (2000)
4. Wang, L., Seungho, J., Lee, T.: Redundant source routing for real-time services in ad hoc networks. In: IEEE International Conference on Mobile Ad Hoc and Sensor Systems Conference 2005, pp. 80–87. IEEE Press, Washington, DC (2005)

5. Peng, Y., Song, Q., Yu, Y., Wang, F.: Fault-tolerant routing mechanism based on network coding in wireless mesh networks. Journal of Network and Computer Applications 37, 259–272 (2014)
6. Nicolaou, N., See, A., Xie, P., Cui, J.H., Maggiorini, D.: Improving the robustness of location-based routing for underwater sensor networks. In: OCEANS 2007-Europe, pp. 1–6. IEEE, Aberdeen (2007)
7. Li, X., Cuthbert, L.: On-demand node-disjoint multipath routing in wireless ad hoc networks. In: 29th Annual IEEE International Conference on Local Computer Networks. IEEE Press, Florida (2004)
8. Draves, R., Padhye, J., Zill, B.: Routing in multi-radio, multi-hop wireless mesh networks. In: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, pp. 114–128. ACM, New York (2004)
9. Pathak, P.H., Dutta, R.: A survey of network design problems and joint design approaches in wireless mesh networks. Communications Surveys & Tutorials 13, 396–428 (2011)
10. Simulator, Q.N.: Scalable Network Technologies, `http://www.qualnet.com`
11. Ikenaga, T., Tusbochi, K., Nobashi, D., Fukuda, Y.: Disjoint Path Routing for Multi-channel Multi-interface Wireless Mesh Network. International Journal of Computer Networks & Communications (IJCNC) 3(2), 165–178 (2011)
12. Shi, W., Ikenaga, T., Nobayashi, D., Yin, X., Xu, H.: Reliable Transmission with Multi-path and Redundancy for Wireless Mesh Networks. In: WNM 2014. Springer, Dalian (2014)

Distributed Efficient Node Localization in Wireless Sensor Networks Using the Backtracking Search Algorithm

Alan Oliveira de Sá[1], Nadia Nedjah[2], and Luiza de Macedo Mourelle[3]

[1] Center of Electronics, Communications and Information Technology
Admiral Wandenkolk Instruction Center, Brazilian Navy
[2] Department of Electronics Engineering and Telecommunication
[3] Department of System Engineering and Computation,
Engineering Faculty, State University of Rio de Janeiro, Brazil
alan.oliveira.sa@gmail.com, {nadia,ldmm}@eng.uerj.br

Abstract. The localization problem arises from the need of nodes of a wireless sensors network to determine their positions without the use of external references, such as the Global Positioning System – GPS. In this problem, the node location may be established thanks to distance measurements to existing reference nodes. Reference nodes know their respective positions in the network. In the search for efficient yet accurate methods to determine node locations, some bio-inspired algorithms have been explored. In this sense, targeting a more accurate solution of the localization problem, we propose a new multi-hop method based on the Backtracking Search Algorithm. It includes a new technique to assess the confidence that should be granted to a contribution received from a neighboring node, and hence incorporating it into the localization computation accordingly. The achieved performance results prove the effectiveness of the proposed method as well as the efficiency entailed by the confidence factor assessment technique. The impact of the latter is more evident when the number of reference nodes in the network is reduced. This constitutes a very big advantage with respect to state-of-the-art localization methods.

1 Introduction

Wireless Sensor Networks (WSNs), whose prospect of application is very broad, have attracted great attention from industry [1]. However, in most cases, WSNs have little use when no positioning sensors are included into the nodes [12]. Similarly, many applications of Swarm Robotic Systems (SRSs) require that a robot is able to discover its position. This position may be either absolute, *i.e.* with respect to a universal reference system, or relative to other robots, based on a local coordinate system.

In both cases of WSNs and SRSs, the basic devices, *i.e.* sensors or robots, respectively, have common characteristics, which are of a reduced size, have access to a limited energy source and must be of low cost. The straightforward solution that consists of endowing each basic device with a Global Positioning System – GPS is often not feasible.

The localization problem consists of inferring the position of a set of sensors or robots when no external reference, such as GPS, is available. Many of the localization methods depend on the ability of a node (sensor or robot) to measure its distance to some reference nodes, also known as anchors. The position of reference nodes are known [10].

In general, common techniques for distance measurement are based on the power of the received signal, its propagation time and/or the comparison of the propagation time of two signals that are known to have different propagation speeds [8,9]. As the measurement techniques presented rely on signal propagation characteristics, a threshold distance for such measurements has to be considered. In the simple case, wherein all reference nodes are within the distance measurement threshold, the measurements are direct and thus, made via a single hop. However, in the cases where one or more reference nodes are outside this threshold, distance measurements are obtained indirectly using a multi-hop strategy. For this purpose, algorithms such as *Sum-dist* or *DV-hop* are used [7]. Depending on the network topology in WSNs or swarm connectivity in MRSs, the use of one and multi-hop can be combined. In [7], a three-step approach is proposed:

1. Estimate the distances of each node to the reference nodes.
2. Compute the position of each node using the measurements obtained in step 1.
3. Refine the position of each node using the positioning and distance information informed by the neighboring nodes.

The mentioned three-step approach is identified in algorithms that present skills of self-organization, robustness, and energy efficiency [7], such as the algorithms: *Ad-hoc positioning* [10], *N-hop multilateration* [13] and *Robust positioning* [11]. Also, this approach provides two advantages: it is possible to analyze partial results in order to have a better understanding of the combined behavior; and it is possible to further mix-and-match other alternatives for both phases, in order to obtain better performances [7].

The optimization using bio-inspired techniques are often applied to the localization problem, in the case of a single hop as well as in that of multi-hop [12,4]. This paper proposes a new method for solving the localization problem based on a multi-hop strategy. The method exploits an optimization process using the Backtracking Search Algorithm – BSA. The methods acts in three stages: During the first stage, the Sum-Dist [7,11] is used to estimate the node distances to the reference nodes; Then, during the second stage, an initial position estimate is made using the Min-Max method [7,11]; After that, during the third stage, the refinement of the positions is performed by mean of an optimization process using BSA. Aiming at improving the evaluation of the accuracy of inferred localizations, we propose and evaluate a new technique to establish the confidence factor, which is used to assess the importance of the contributions received from neighboring nodes.

The rest of this paper is organized as follows: First, in Section 2, we present some related works. Later, in Section 3, we briefly describe the main steps of BSA; In Section 4, we show the proposed distributed localization method together with the novel technique to establish the confidence factor of the feedback made by the neighboring nodes; Then, in Section 5, we report and discuss on the achieved performance results. Finally, in Section 6, we present some concluding remarks along with some possible future work.

2 Related Work

The importance of the localization information to sensors in wireless networks and robots of a swarm, conjugated with the limitations in terms of hardware and energy requirements that are typical of these devices, have motivated the search for more efficient yet accurate algorithms to solve the localization problem.

In [12], the authors report on the use of genetic algorithms for solving the problem of localization in sensor networks without barriers and without noise.

Another approach to the localization problem is presented in [6]. The authors propose the use of genetic algorithms to determine the position of unknown static nodes. In contrast with [12], this localization was done with respect to mobile reference nodes. This application showed to be inefficient for networks that are spread through a large coverage area. Mainly, this drawback was due to the high-energy required during the movement of the reference nodes.

In [4], the author presents a localization algorithm based on swarm intelligence. I uses Particle Swarm Optimization – PSO to evolve a solution. The work is applicable to network of static sensor nodes only. However, the extension of the proposed algorithm to mobile sensor networks was later demonstrated in [5]. Two or three dimensions can be considered. The distance measurements to reference nodes are performed using a multi-hop strategy.

3 Backtracking Search Algorithm

The Backtracking Search Algorithm – BSA is a relatively new evolutionary algorithm [2] that uses knowledge gathered from past generations to seek solutions of better fitness. The bio-inspired philosophy behind BSA is similar to that of a social group of animals that, at random intervals returns to hunting areas that were previously visited for food foraging. The general structure of BSA is shown in Algorithm 1.

During the initialization step, the algorithm generates and evaluates the initial population P_0 and sets the historical population P_{hist}. The latter composes the BSA's memory.

During the first selection step, the algorithm randomly determines, based on a uniform distribution, whether the current population P should be kept as the new historical population, and thus replace P_{hist}. Subsequently, it shuffles the individuals of this population.

Algoritmo 1. BSA

```
begin
    Initialization;
    repeat
        Selection-I;
        Generate new population
            Mutation;
            Crossover;
        end
        Selection-II;
    until Stopping Condition;
end
```

The mutation operator creates P_{mod}, which is the preliminary version of the new population P_{new}). It does so according to (1):

$$P_{mod} = P + \eta \cdot \Gamma(P_{hist} - P), \tag{1}$$

wherein η is adjusted empirically via simulations and $\Gamma \sim N(0,1)$, with N being a normal standard distribution. Thus, P_{mod} is the result of the movement of P's individuals in the directions established by vector $(P_{hist} - P)$.

In order to create the final version of P_{new}, the crossover operator combines randomly, also following a uniform distribution, individuals from P_{mod} and others from P.

In second selection stage, the algorithm evaluates, selects elements of P_{new} (*i.e.* individuals obtained after mutation and crossover), which should have better fitness than those in P (*i.e.* individuals before applying both the operators of crossover and mutation) and replaces them in P. Hence, P includes only new individuals that should have evolved. While the stopping condition has not yet been reached, the algorithm iterates. Otherwise, it returns the best solution found.

4 Localization Proposed Method

The localization proposed method can be applied to problems with two or three dimensions. However, for the sake of clarity of the analysis presentation, but without loss of generality, we focus on the case of two dimensions. As a premise, the problem is regarded as multi-hop. This means that the unknown nodes, in most cases, cannot measure their distance to the reference node directly. In this case, the node estimates it distance to the anchors via an available multi-hop path. In this formulation, all nodes are considered static and no distance measurement errors are taken into account. Also, let A be the search space, V_i the set of neighboring nodes of node i and R the set of the reference nodes of the network.

Like the approach presented in [7], as reported in the introduction, the proposed method operates in three stages. In the remainder of this section we introduce the underlying details of each of these stages; then we explain the new technique to establish the confidence factor associated with the position transmitted by a neighboring node. This factor is used in the third stage of the algorithm; subsequently, we present the complete algorithm of the proposed method. It is worth noting that all the nodes of the network execute the same localization algorithm simultaneously. Furthermore, for a certain period of time at the beginning of the localization process, all the network reference nodes keep sending messages composed by their respective position together with the message traveled distance (0 at this moment), to all neighboring nodes.

4.1 Distance to Reference Nodes

Mainly, the first stage (STAGE-I) allows for node i to estimate its distance to the reference nodes using the Sum-Dist algorithm [7]. Sum-Dist algorithm is chosen to perform STAGE-I due to its low computation and communication costs [7]. This stage starts by disseminating any message that was transmitted by an anchor. The message is passed from node to node, so that the distance in each hop is added to the total distance traveled by the message. The message received by a node is stored and then retransmitted if and only if, the distance traveled by the message is the so-far smallest distance to the given reference node. Thus, each unknown node i will be aware of the position coordinates ($\hat{x}_r$, $\hat{y}_r$) of all the network reference node $r \in R$ and the distance of the shortest route $l_{i,r}$ that exists between them.

Note that $l_{i,r}$, *i.e.* the shortest distance traveled by a message from unknown node i till anchor r, is not necessarily the actual distance between them. For instance, in a scenario with randomly distributed nodes, it is common that nodes that compose the path between i and r are not aligned, causing that $l_{i,r}$ is greater than the actual distance between the two nodes.

4.2 Initial Node Position

The second stage (STAGE-II) allows for node i to estimate its initial position u_i. This is done using the Min-Max technique [7,11]. It identifies for each reference node r a region, denominated as *bounding box* $B_{i,r}$, whose boundaries are computed by adding and subtracting $l_{i,r}$ from the position of each reference node ($\hat{x}_r$, $\hat{y}_r$), according to (2). Recall that both $l_{i,r}$ and ($\hat{x}_r$, $\hat{y}_r$) were obtained during STAGE-I.

$$B_{i,r} : [\hat{x}_r - l_{i,r}, \hat{y}_r - l_{i,r}] \times [\hat{x}_r + l_{i,r}, \hat{y}_r + l_{i,r}]. \tag{2}$$

Let S_i be the region corresponding to the intersection of all these bounding boxes as defined in (3):

$$S_i = \bigcap_{\forall r} B_{i,r} : [\max_{\forall r}(\hat{x}_r - l_{i,r}), \max_{\forall r}(\hat{y}_r - l_{i,r})] \times [\min_{\forall r}(\hat{x}_r + l_{i,r}), \min_{\forall r}(\hat{y}_r + l_{i,r})]. \tag{3}$$

The initial position u_i of unknown node i is computed as the center point of S_i. Fig. 1 illustrates the Min-Max technique to estimate the position of an unknown node i from three anchors r_1, r_2 and r_3.

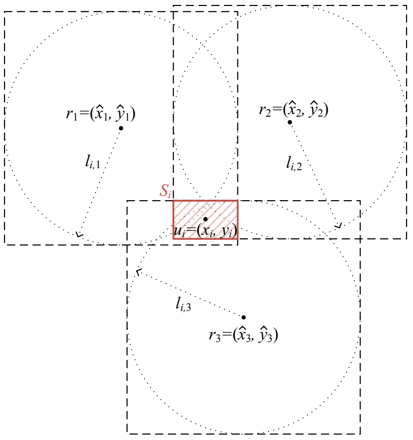

$r_1=(\hat{x}_1, \hat{y}_1)$
$l_{i,1}$
$r_2=(\hat{x}_2, \hat{y}_2)$
$l_{i,2}$
S_i
$u_i=(x_i, y_i)$
$l_{i,3}$
$r_3=(\hat{x}_3, \hat{y}_3)$

Fig. 1. Graphical illustration of the Min-Max method

Alternatives to Min-Max consists of using an optimization algorithm, as it was done in [4], or using the Lateration method [7]. However, even with the use of those processes, the computation remains dependent on $l_{i,r}$, which is predominantly inaccurate. Since both methods are impacted by the inaccuracy of $l_{i,r}$, we opted by using the Min-Max approximation for its simplicity. Furthermore, it results in a reduced computational cost when compared to both evolutionary optimization process or Lateration method.

4.3 Node Position Refinement

The third stage (STAGE-III) is an iterative process that achieves an accuracy improvement of the unknown node positions. During this stage, each unknown node re-evaluates its position based on those of its neighbors as well as on the inferred distances to those neighbors. Note that the initial position of the unknown nodes are those computed during STAGE-II. These positions are updated at each iteration. Since, ideally the measures of distance between nodes remain constant, the estimated positions of the unknown nodes tend to be adjusted gradually, in order to finally and hopefully, converge to their real positions.

In this stage, the position refinement is performed using BSA. The objective function is a composite of three main terms. For a given unknown node i, we first consider the error introduced in the distance to each of its neighboring nodes $v \in V_i$ [4]. In this case, node v is considered a neighbor of node i, if and only if it is accessible from i via one single hop. Note that node v may either be an anchor or an unknown node. The considered error is defined in (4):

$$g(i) = (d_{i,v} - ||p_v - p_i||)^2 , \tag{4}$$

wherein $d_{i,v}$ is the estimated distance between unknown node i and its neighbor node v, p_v is the position inferred by v and p_i is that of node i.

The first term of the objective function takes into account the error introduced by each unknown node i, and normalizing it by a confidence factor that establishes how accurate is the knowledge passed by neighbor $v \in V_i$. Hence, this term is defined as in (5):

$$f_1(i) = \sum_{v \in V_i} \frac{g(i)}{\zeta_v}, \tag{5}$$

wherein ζ_v is the confidence factor associated with neighbor v. The technique used to assess the confidence factor of a node is detailed in 4.4.

The second term of the objective function considers those nodes that are two hops away from unknown node i. According to [4], the distance between unknown nodes i and w, that is positioned at two-hop distance away from each other, is always greater than the threshold of distance measurement, denominated L, but always smaller than $2L$. Thus, the contribution of node w in the second term of the objective function can be defined as in (6):

$$h(i) = \max(0, L - ||p_w - p_i||, ||p_w - p_i|| - 2L)^2, \tag{6}$$

wherein p_w is the position of node w. Observe that if a possible solution is such that $L \leq ||p_w - p_i|| \leq 2L$, then no value should be added to its fitness. However, if $||p_w - p_i|| < L$, then the square of the distance between p_i and the circumference of radius L, centered at p_w should be added to the fitness. Otherwise, *i.e.* if $2L < ||p_w - p_i||$, then the square of the distance between p_i and the circumference of radius $2L$, centered at p_w should be added to the fitness.

Based on the aforementioned explanation, the second term of the objective function is defined as in (7):

$$f_2(i) = \sum_{w \in W_i} \frac{h(i)}{\zeta_w}, \tag{7}$$

wherein W_i is the set of nodes that are two hops away from i and ζ_w is the confidence factor related to the knowledge informed by node w. Its definition is detailed in Section 4.4.

The third term of the objective function, defined as in (8), is intended to guide the possible solutions towards area S_i, as defined by the Min-max technique, during second stage of the proposed method (see Section 4.2). Thus, if a possible solution falls within S_i, then no value should be added to its fitness. Otherwise, *i.e.* if it is situated outside S_i, then a value that is proportional to the square of the distance between the solution and the center of S_i should be added to its fitness.

$$f_3(i) = \begin{cases} 0 & \text{if } p_i \in S_i; \\ \varphi_1 + \varphi_2 ||p_i - u_i||^2 & \text{if } p_i \notin S_i. \end{cases} \tag{8}$$

wherein u_i is the position of the center of S_i and φ_1 and φ_2 are empirically set positive parameters. They allow to always favor positions inside S_i against those outside even though they are both very close the border of S_i.

Finally, the objective function is then given by (9):

$$\min_{p_i \in A} f(i) = f_1(i) + f_2(i) + f_3(i), \tag{9}$$

wherein p_i is the position of node i and A is the whole search space. Note that, as there are no errors in distance measurements, the global minimum value of $f(i)$ is 0 for any node i.

4.4 Confidence Factor

The confidence factor evaluates the degree of accuracy of a given position as informed by a neighboring node. It gives more importance to a node contribution that tends to have more accurate positions. In this sense, a position is said to be more accurate than another if the former is closer to the anchors than the latter. Another way to assess the accuracy of a position is how confined is region S_i (as defined in Section 4.1). Recall that this factor is used in (5) and (7). Two different methods are used to evaluate this confidence factor. It is worth noting that in both views, the confidence factor of an anchor must be 1.

The first technique [4], denominated as *Hops to Anchor Confidence Factor* – (HTA-CF), is based on the idea that the closer the nodes are to the anchors the more accurate their positions should be. In this method, the confidence factor is calculated according to (10):

$$\zeta_i^{(\mathrm{HTA})} = \begin{cases} 1 & \text{if } i \in R \\ \sum\limits_{r \in R_i} \lambda_{i,r}^2 & \text{if } i \notin R. \end{cases} \tag{10}$$

wherein R_i is the set composed by the three closest reference nodes to i in terms of total number of hops, and $\lambda_{i,r}$ is the number of hops between node i and anchor r.

The second technique, proposed herein, designated as *Min-Max Area Confidence Factor* – (MMA-CF), takes into account the area S_i, established by the Min-Max method. It corresponds to the region where unknown node i must be positioned [11]. Thus, it can be inferred that the smaller the region is, the greater the possibility of position u_i, established in the second stage (see Section 4.2), is closer to the actual position.

Depending on the topology of the network, node i that is far away from any reference node may have a reduced area S_i, and consequently, has a good reliability. This may occur, for instance, when there is a good alignment of the nodes that are between i and reference nodes. In this case, distances $l_{i,r}$ from i to any reference node $r \in R$ are very close to the actual distances, hence, reducing the size of S_i. The confidence factor MMA-CF for a node i is defined as (11):

$$\zeta_i^{(\mathrm{MMA})} = \begin{cases} 1 & \text{if } i \in R \\ \\ 1 + S_i & \text{if } i \notin R. \end{cases} \tag{11}$$

Fig. 2 shows the behavior of the confidence factors HTA-CF and MMA-CF for a network that includes 10 anchors and 200 unknown nodes. In this figure, the nodes were classified in an ascending order according to the value of HTA-CF. Observe that some nodes that are associated with a high confidence factor, according to HTA-CF, are distant from the reference nodes of the network. These have a low confidence factor according to MMA-CF, *i.e.* have a small area S_i. Some of these nodes are highlighted in Fig. 2. This indicates that, even when far away from the reference nodes, these nodes tend to have a "good" initial positions with respect to accuracy, as computed during the second stage of the proposed method (see Section 4.2). Hence, these nodes can be considered, at least in the initial iterations of the third stage, as nodes with "good" reliability, even though these are away of the reference nodes. This property of MMA-CF suggests the possibility of improving the performance of the optimization process used during the third stage of the proposed method. Further evaluations of the impact of MMA-CF *vs.* HTA-CF were conducted. The results are reported in Section 5.

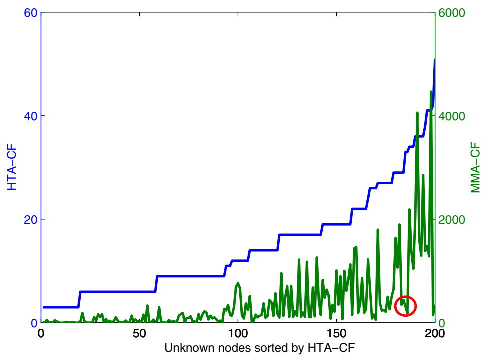


Fig. 2. Comparison of HTA-CF *vs.* MMA-CF for a network of 10 reference nodes and 200 unknown nodes

4.5 Algorithm of the Proposed Method

The complete algorithm of the proposed method to the multi-hop localization problem is shown in Algorithm 2. It includes the three main stages, as presented previously. Note that the algorithm is executed simultaneously by each unknown node of the network, yielding the emergence of the localization of all the network nodes. In Algorithm 2, $\varDelta$ defines the number of cycles performed during the third stage while δ defines the number of the BSA iterations, done in each of these cycles.

Algoritmo 2. Complete algorithm of the proposed method at node i

```
begin
    STAGE-1: Sum-Dist
        Message exchange to get $(\hat{x}_r, \hat{y}_r)$ and $l_{i,r}$ for $r \in R$, using Sum-Dist;
    end

    STAGE-2: Min-Max
        Compute $u_i$ using Min-Max;
        Compute the confidence factor $\zeta_i$;
        Message exchange to share $u_i$ and $\zeta_i$ with all neighbors $v \in V_i$;
    end

    STAGE-3: BSA minimization
        Initialize $P$ and $P_{hist}$;
        Insert $u_i$ into $P$;
        Compute fitness of $P$ using $(p_v, \zeta_v)$ of all 1-hop neighbors $v \in V_i$ and
        $(p_w, \zeta_w)$ of 2-hop neighbors $w \in W_i$, as received from STAGE-II;
        for $1 \leq j \leq \Delta$ do
            if new $p_v$ and $p_w$ received then
                Update fitness of $P$ using $(p_v, \zeta_v)$ of 1-hop neighbors $v \in V_i$
                and $(p_w, \zeta_w)$ of 2-hop neighbors $w \in W_i$, as received from
                STAGE-III;
            end
            for $1 \leq k \leq \delta$ do
                Selection-I;
                Generate new population
                    Mutation;
                    Crossover;
                end
                Selection-II;
            end
            Exchange messages to share new $p_i$ and new $\zeta_i$;
        end
    end
    return Best result;
end
```

5 Performance Results

In order to evaluate the performance of the proposed method, we conducted simulations in an area $A = 100 \times 100$ measurement units. In this search space, 100, then 150, and then 200 unknown nodes were randomly distributed. Respectively, we randomly allocated 10, then 20, and then 30 reference nodes. The distance measurement limit L was set to 20 units. For each combination of the reference and unknown nodes, 10 scenarios were generated, resulting in a total of 90 different simulations.

To evaluate the proposed method together with the performance of the new technique used to compute the confidence factor, two versions of the algorithm were tested: one using HTA-CF and the other MMA-CF. It worth emphasizing that, although randomly generated, the same scenarios were precisely reproduced and submitted to both algorithm versions, thus guaranteeing the fairness of the performance comparison. In all simulations, we employed a BSA with a population of 100 individuals.

The proposed algorithm was implemented using MATLAB. As a basis, we used the implementation of BSA available in [3]. Parameter η used by BSA was adjusted empirically through several simulations. The best results were obtained with $\eta = 1$. The values of Δ and δ, of the third stage, were set to 100 and 10, respectively. Moreover, the objective function parameters φ_1 and φ_2 were set to 10^3 and 10^6, respectively.

For the purpose of the comparison of the impact provided by the confidence factors, we computed, for each iteration, the mean positioning error (*MPE*) of all unknown nodes, as defined in (12):

$$MPE_j^{(s)} = \frac{\sum_{i=1}^{I} ||p_{real}^{(s)}(i) - p_{comp}^{(s)}(i)||}{I}, \tag{12}$$

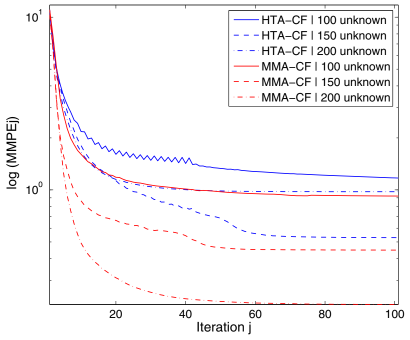
wherein j represents the number of the iteration during the third stage, i the unknown node, I is the total number of unknown nodes, p_{real} is the actual position of the node and p_{comp} is the estimated position by the optimization process, as achieved until iteration j.

In order to synthesize all the $MPEs$ of the simulated 10 scenarios, generated for each pair of the numbers of reference and unknown nodes, $MMPE$, which represents the average for all $MPEs$, was computed using (13):

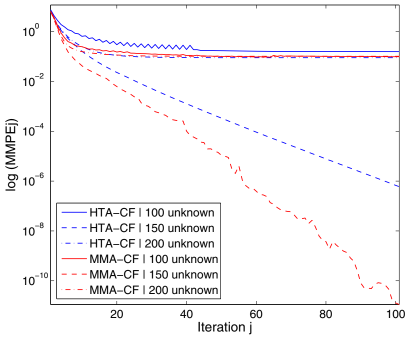
$$MMEP_j = \frac{\sum_{s=1}^{10} MPE_j^{(s)}}{10} \tag{13}$$

The results of $MMEP$ per iteration of the third stage, obtained by the two versions of the algorithm, are depicted in Fig. 3(a) for the combination of 100, 150 and 200 unknown nodes with 10 anchors. Fig. 3(b) shows the same results for the combination of 100, 150 and 200 unknown nodes with 20 anchors. Fig. 3(c) shows the results for the combination of 100, 150 and 200 unknown nodes with 30 anchors.

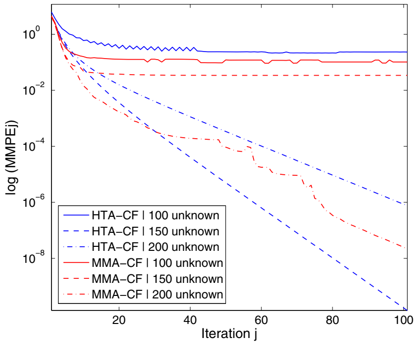
Considering both versions of the algorithm, we can see that in the worst case, *i.e.* with 10 reference nodes and 100 unknown nodes, the $MMEP$ is approximately 1 measurement unit and in the best case, *i.e.* with 30 reference nodes and 200 unknown nodes, the $MMEP$ reached the order of 10^{-10} measurement units. Recall that, this is with respect to a total search area of 100×100 measurement units. In general, this proves the effectiveness of the proposed method, independently of using either of the confidence factors. Note that the accuracy of the solution tends to increase with the number of reference nodes. Likewise, an increase in connectivity, *i.e.* when the number of unknown nodes increases, also contributes positively to the accuracy of the inferred positions.



(a) 10 anchors



(b) 20 anchors



(c) 30 anchors

Fig. 3. The behavior of MMPE × iteration, for the network different configuration

A summary comparison of the techniques based on HTA-CF and MMA-CF is shown in Table 1, wherein for each combination of the numbers of reference and unknown nodes, it indicated the confidence factor that has achieved the best performance after 100 iterations.

Table 1. A comparative summary of HTA-CF and MMA-CF

	100 unknown	150 unknown	200 unknown
10 anchors	**MMA-CF**	**MMA-CF**	**MMA-CF**
20 anchors	**MMA-CF**	**MMA-CF**	**HTA-CF**
30 anchors	**MMA-CF**	**HTA-CF**	**MMA-CF**

The reported results indicate that the MMA-CF, in most cases, provided better performance than HTA-CF, especially when there were fewer reference nodes in the network. It is also possible to observe in Figure 3(b) and Figure 3(c) that even in the cases where the HTA-CF is more efficient, MMA-CF presents

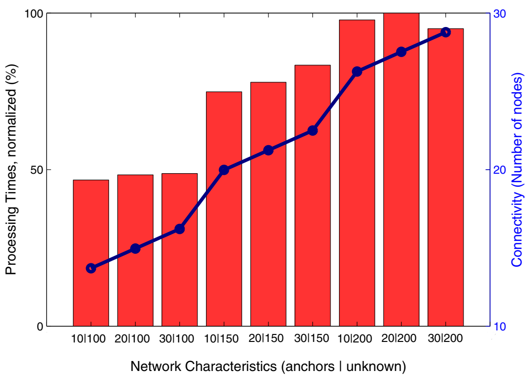


Fig. 4. Comparison of processing time *vs.* connectivity

lower values of MMEP during the initial cycles of the execution. In these cases, the performance of the MMA-CF, observed at the beginning of the optimization process together with the good performance of HTA-CF, during the subsequent cycles, points out that, in order to further improve both accuracy and efficiency, both both factors should be combined.

An assessment of the processing time of is shown in Fig. 4, where it is possible to relate the time increase to the increase in terms of network connectivity. Network connectivity is defined as in (14):

$$connectivity = \frac{(I + R - 1)\pi L^2}{A} \tag{14}$$

wherein R is the total reference nodes and A is the total search space area. Recall that L is the measurement threshold and I is the total number of unknown nodes in the network.

It is noteworthy to point out that there was no significant difference in terms of processing time with respect to the used confidence factor. This is simply due to the fact that both factors underlying computation represents a very small portion of the overall computational effort required by the entire localization process.

6 Conclusions

Based on the presented results, we can conclude that the proposed method to solve the localization problem is effective, with both confidence factors: HTA-CF and MMA-CF. Such effectiveness is proven by the accuracy achieved. In the worst case, the mean error introduced is of the order of 1 measurement unit, while in the best case, it is of the order of 10^{-10} measurement units. The measured whole area is of 100×100 measurement units.

The proposed technique, to compute the confidence factor MMA-CF, performs very well, particularly when the number of anchors is reduced, which is almost always the case in real world applications. The performance speedup obtained, mainly at the beginning of the optimization process, when using MMA-CF, and the best performance observed in subsequent cycles with HTA-CF, for some network configurations, encourages the combination of both factors to take advantage of the nice features of both. This points out one of the possible near future investigation.

Also, as future work, we propose to investigate the performance of the method proposed herein when there is noise in the distance measurements process. This will be implemented using a swarm of real robots. Another interesting investigation is to assess the performance of the proposed method when in localization problems that involve node mobility.

Acknowledgement. We thank the State of Rio de Janeiro (FAPERJ, http://www.faperj.br) for funding this research work.

References

1. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Ad Hoc Network Journal 3, 325–349 (2005)
2. Civicioglu, P.: Backtracking Search Optimization Algorithm for numerical optimization problems. Applied Mathematics and Computation 219, 8121–8144 (2013)
3. Civicioglu, P.: BSA MATLAB code, `http://www.pinarcivicioglu.com/bsa.html` (accessed December 11, 2013)
4. Ekberg, P.: Swarm-Intelligent Localization, Thesis, Uppsala Universitet, Uppsala, Sweden (2009)
5. Ekberg, P., Ngai, E.C.: A Distributed Swarm-Intelligent Localization for Sensor Networks with Mobile Nodes. In: 7th Int. Wireless Communications and Mobile Computing Conference, pp. 83–88 (2011)
6. Huanxiang, J., Yong, W., Xiaoling, T.: Localization Algorithm for Mobile Anchor Node Based on Genetic Algorithm in Wireless Sensor Network. In: Proc. International Conference on Intelligent Computing and Integrated Systems, pp. 40–44. IEEE (2010)
7. Langendoen, K., Reijers, N.: Distributed Localization Algorithms. In: Zurawski, R. (ed.) Embedded Systems Handbook, pp. 36.1–36.23. CRC Press (2005)
8. Lymberopoulos, D., Lindsey, Q., Savvides, A.: An empirical characterization of radio signal strength variability in 3-D IEEE 802.15.4 networks using monopole antennas. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 326–341. Springer, Heidelberg (2006)
9. Mao, G., Fidan, B., Anderson, B.: Wireless sensor network localization techniques. Computer Networks 10 51, 2529–2553 (2007)
10. Niculescu, D., Nath, B.: Ad hoc positioning system (APS). In: Proc. IEEE Global Telecommunications Conference, GLOBECOM 2001, pp. 2926–2931 (2001)
11. Savvides, A., Park, H., Srivastava, M.B.: The bits and flops of the n-hop multilateration primitive for node localization problems. In: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, pp. 112–121. ACM (2002)

12. Sun, W., Su, X.: Wireless sensor network node localization based on genetic algorithm. In: Proc. 3rd International Conference on Communication Software and Networks, pp. 316–319. IEEE (2011)
13. Savarese, C., Langendoen, K., Rabaey, J.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: USENIX Technical Annual Conference, Monterey, CA, pp. 317–328 (2002)

User Specific QoS and Its Application in Resources Scheduling for Wireless System

Chao He and Richard D. Gitlin

Department of Electrical Engineering
University of South Florida
Tampa, Florida 33620, USA
chaohe@mail.usf.edu, richgitlin@usf.edu

Abstract. This paper describes user specific QoS requirements that are a critical innovation for improving spectral utilization for wireless systems. An adaptive scheduler is presented that incorporates user specific QoS requirements in the spectral allocation of resources. In this paper, we focus on voice applications, and demonstrate that by dynamically adapting MAC scheduling algorithms to the user specific QoS requirements, user satisfaction, as measured by the user specific Mean Opinion Score (MOS), is maximized. OPNET LTE system simulations have been performed for a set of AMR VoIP users with assigned specific QoS target levels. Simulation results show that significant MOS improvements can be achieved if such user specific QoS requirements are considered in the MAC scheduler. Furthermore, when targeted to maximize spectrum utilization and combined with AMR codecs matched to the auditory characteristics of users, higher system capacity, at comparable MOS levels, may be achieved.

Keywords: User specific QoS, Cross layer scheduler, MOS, AMR.

1 Introduction

In today's wireless 4G LTE networks, the spectral allocation of resources is either independent of the application's specific Quality of Service [QoS] requirements and of the users' specific perceived QoS, or at most relies on a set of pre-defined fixed priorities[1, 2]. Although in these standards, the MAC and the PHY layers have an increased role in optimizing the usage of the spectral resources and implementing link quality-aware techniques, nevertheless, optimization is still largely independent of the application context, the users' requirements, and the users' perception of performance degradation. In particular, the standards do not take into account the Quality of Service (QoS) required by different applications and their users, beyond simply assigning fixed priorities to traffic classes. Indeed, from the user's perspective, the QoS required by different applications can be quite variable. Similarly, for a given application type, different users may require different levels of QoS.

As a motivating example, consider the fact that the perceived voice quality of different languages may differ substantially when allocated the same data rate and the same Bit Error Rate (BER), because of the different spectral content of such languages and because of a particular user's auditory spectral response (with variations typically due to aging), making the user more or less sensitive to a particular type of distortion. Consequently, the same amount of degradation, as experienced by individual applications and their users, may have substantially different perceptual effects. Another example is the varying talk environments, where some users have a conversation under very noisy conditions, while some other users converse under very quiet conditions, thus making users more or less sensitive to packet losses. If the same amount of spectral resources is allocated to users in very noisy and quiet backgrounds, then a highly degraded user experience will likely be incurred in the noisy environment. As another example, consider that people from different age groups normally have different sensitivity to high frequency content[3], which can be exploited to maximize the system capacity by reducing the bit rate for users with reduced frequency sensitivity.

Furthermore, we observe that some previous studies (e.g. [4–7]), which use the QoS characteristics of an underlying application (typically expressed as a function of the Mean Opinion Score [MOS]), allocate average spectral resources to applications, independently of the application's actual specific QoS requirement. Though, in the literature, there are MAC schedulers that take into account instantaneous data rates and user's QoS [8, 9], to date no user-specific QoS requirements have been considered in the MOS functions and in the MAC scheduler. Thus, in such schemes, especially for applications with widely varying QoS requirements (even for the same type of application), either the spectral resources are not efficiently utilized or the MOS is significantly degraded.

Based upon the user specific requirements, in this paper, we will derive a user specific MOS formula and present a novel user specific QoS-aware cross-layer scheduler that maximizes user satisfaction (MOS) through dynamically adapting MAC scheduling algorithms to these user specific QoS requirements. Here, we focus on voice applications in the context of 4G LTE wireless systems. Moreover, we also address improving system capacity by observing that some users are less sensitive to the high frequency content.

The paper is organized as follows. In Section 2, the VoIP E-Model algorithm is described. A brief summary of prior work on the MAC scheduler is presented in Section 3. In Section 4, our user specific MOS formula is derived and user specific QoS aware scheduling approach is described. Section 5 presents our user specific frequency sensitivity research. Section 6 presents the OPNET LTE system simulation setup. In Section 7, the system simulation results of the user specific QoS scheduler are shown. Finally, our conclusions and future research directions are presented in Section 8.

To summarize, it is the purpose of this paper to introduce and evaluate the performance of an adaptive scheduler that incorporates user-specific QoS requirements in the spectral allocation of resources to optimize the MOS and/or the system capacity.

2 E-Model Algorithm

The E-Model algorithm [10] is a computational model for objective call quality assessment, is described in the G.107 recommendation by the ITU-T. The computation of the MOS is defined as follows:

$$\mathrm{R} = R_0 - I_d - I_{eff} \tag{1}$$

where R is the transmission rating factor, which combines all transmission parameters relevant for the considered connection. R_0 is the basic signal-to-noise ratio which has a default value of 93.2 [11, 12], I_d represents the impairments due to delay, which is the same for all the codec modes, and I_{eff} represents the effect of packet losses and depends on the codec (e.g. AMR, G.711) that is used.

$$I_d = 0.024d + 0.11(d - 177.3)U(d - 177.3) \tag{2}$$

where d is the end-to-end delay in milliseconds and U is the unit step function [12].

For AMR codecs [10],

$$I_{eff} = I_e + (95 - I_e)\left(\frac{100P_{pl}}{\frac{100P_{pl}}{BurstR} + B_{pl}}\right) \tag{3}$$

where P_{pl} represents packet loss ratio, $BurstR$ is the Average length of observed bursts in an arrival sequence to the Average length of bursts expected for the network under "random" loss ratio. In this paper we assume the packet loss is independent and hence we set $BurstR = 1$. B_{pl} is the robustness factor which is set to 10 for all AMR codec modes. I_e is defined for all AMR codec modes in[13], where eight AMR-NB codec modes are defined in LTE [14].

For G.711 codecs [12],

$$I_{eff} = 0 + 30\ln\left(1 + 15P_{pl}\right) \tag{4}$$

R is converted to MOS according to (5):

$$\mathrm{MOS} = \begin{cases} 1, & when\ R < 0 \\ 1 + 0.035R + R(R-60)(100-R) & \\ \cdot 7 \cdot 10^{-6}, & when\ R \in [0, 100] \\ 4.5, & when\ R > 100 \end{cases} \tag{5}$$

3 Current MAC Scheduler Approaches

3.1 The MAC Scheduler

The MAC Scheduler is a key component of the LTE Evolved NodeB (eNodeB). The function of the scheduler is to facilitate the allocation of the available spectral resources (e.g., time and frequency resources), while striving to satisfy the QoS requirement of all the users.

Two of the main functions of the LTE radio scheduling are dynamic packet scheduling and link adaptation [8, 9], where the scheduler needs the input of the link adaptation module to select the appropriate Modulation and Coding Scheme(MCS) for channel dependent scheduling. In dynamic packet scheduling, the time-frequency domain resources are distributed dynamically among the active users to get their packets scheduled at the MAC layer. The packet scheduling comprises two scheduling components [8, 9]. They are done sequentially in each scheduling time unit, known as Transmission Time Interval (TTI) in LTE (TTI = 1ms). The first component is the time domain scheduler (TDS) and the second is the frequency domain scheduler (FDS). Such a split is driven simply by the consideration of lower complexity and independent configurations for both domains. The objective of the time domain scheduler is to choose a subset of all users requesting frequency resources, while the objective of frequency domain scheduler is to allocate physical resources for the candidate users provided by the time domain scheduler. Several basic scheduling algorithms exist both in time and frequency domains[8][9]:

1. Round-Robin scheduling algorithm

Users are served in a Round-Robin way so that each user is served fairly but at the expense of system throughput and spectral efficiency.

2. Maximum C/I scheduling algorithm

Users with the maximum C/I [Carrier-to-Interference power ratio] are served first. This kind of scheduling aims to achieve maximum benefits in terms of system throughput and spectral efficiency but comes at the expense of fairness.

3. Proportional-fair (PF) scheduling algorithm

PF scheduling algorithm aims to tradeoff the system throughput for the users' fairness. The PF priority metric is calculated by dividing the predicted user's throughput, which is the instantaneous supportable data rate, by the estimation of the user's past average throughput.

3.2 LTE Baseline Scheduler

The benchmark for performance comparison is the LTE baseline scheduler, where the time domain and frequency domain schedulers are as follows:

1. Time Domain Scheduler

Since the VoIP service is a real time service, it is served and scheduled in real time with the highest priority compared with other non-real time services. But, VoIP users can tolerate a certain amount of delay without being scheduled strictly in real time. The pre-defined scheduling delay is set to 80ms in the LTE baseline scheduler.

2. Frequency Domain Scheduler

Each user has a C/I metric for each sub-band in the system bandwidth and is sorted for each sub-band among all the scheduled users. A max C/I approach is used in the LTE baseline scheduler where each sub-band is first allocated to the user that has the highest C/I , then to the user with the second and third highest C/I, and so on until all the resources of this given sub-band are allocated.

4 User Specific QoS Aware Scheduler

The novelty of the proposed cross layer scheduler is that it incorporates the user specific QoS requirements into the scheduling and differentiates the UEs' scheduling utilizing this user specific QoS information to improve system performance as described below. The cross layer scheduler is aware of the individual QoS requirements of the users, including those with the same application type, and uses this information to optimize the scheduling algorithm by giving higher scheduling priority to those users that are more sensitive to the voice quality. One of the differences with user specific QoS requirements addressed by the cross-layer scheduler in this paper is users different sensitivity to packet losses. Another is users different sensitivity to the high frequency content of the speech signal as a function of age and other factors, which is addressed in Section 5.

4.1 UE-Specific MOS Formula

Here we have assumed that different people have similar sensitivity to the end-to-end delay for VoIP applications, so that only UE specific sensitivity to packet losses is studied. To reflect different users sensitivity to packet losses, a UE specific sensitivity factor, α, is added to (1) so that the metric becomes:

$$\mathrm{R} = R_0 - I_d - \alpha \cdot I_{eff} \tag{6}$$

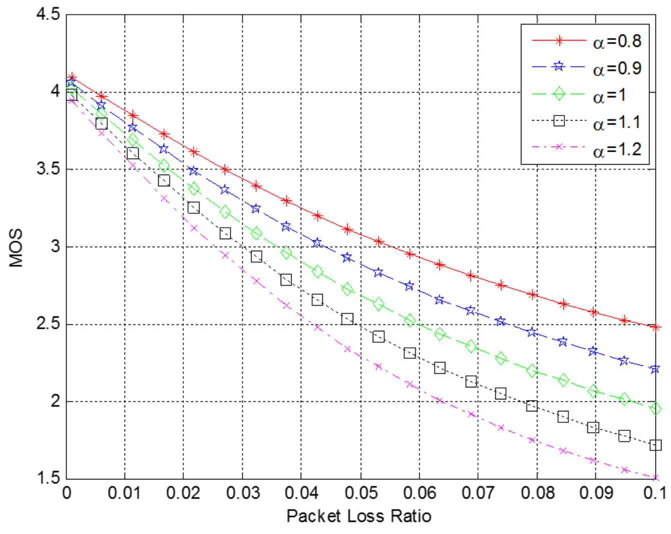


Fig. 1. MOS versus packet loss ratio for different sensitivity factors **α** for AMR 10.2K

In our simulation setup, with the configuration described in Table 1, without loss of generality and also for simplicity of illustration, the sensitivity factor α takes values from the following set $\{0.8, 0.9, 1, 1.1, 1.2\}$. The higher the value of the sensitivity factor (α) is, the user is more sensitive to the packet loss. When α takes the value of 1, it is a normal user. When α takes the value greater than 1, it is more sensitive to packet losses compared with the normal user. When α takes the value less than 1, it is less sensitive to packet losses compared with the normal user. For a given acceptable mouth-to-ear end-to-end delay of 150 ms, Fig. 1 shows the MOS as a function of packet loss ratio for different sensitivity factors **α** for AMR 10.2 Kbps. From Fig. 1, we can clearly see that with the same packet loss ratio and end-to-end delay, different users with different sensitivity factor α will have different MOS values, i.e. a different user experience. As the packet loss ratio becomes larger, the MOS difference also becomes larger. Comparing two extreme cases of α (e.g. α = 1.2 and α = 0.8), a significant difference can be observed for a wide range of packet loss ratios. The user specific QoS-aware scheduler can make use of this information to optimize the scheduling by giving higher priority to users with larger sensitivity factors.

4.2 Optimization Principles

Depending upon the optimization target, there are two kinds of schedulers. One is the MOS targeted scheduler that aims to maximize user satisfaction. The other is the spectral utilization targeted scheduler that aims to maximize system capacity, while maintaining an acceptable MOS level at the same time. In this paper, we focus on the MOS targeted scheduler, while the capacity targeted scheduler is the focus of future work.

With AMR10.2K VoIP users as the example, according to the ITU-T G.107 E-model, the MOS value depends upon both the packet loss ratio and delay. Fig. 2 illustrates the relationship between the MOS value, the delay, and packet loss ratio.

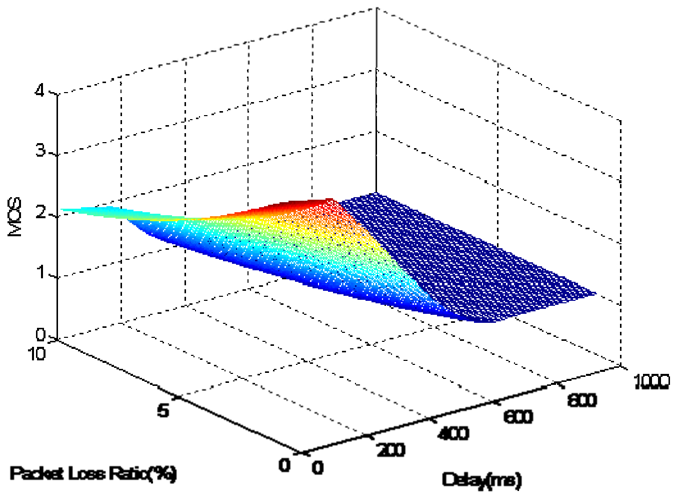


Fig. 2. MOS as a function of packet loss and delay for AMR10.2K

The higher the delay, or the higher the packet loss ratio, the lower the MOS value. Thus, the MOS can be improved either through the optimization of the time domain scheduler or frequency domain scheduler to reduce the delay or packet loss ratio respectively if the user-specific QoS requirement information is known by the scheduler. To be more specific, when a given UE has higher QoS requirement (e.g. higher sensitivity factor), the scheduler can give a higher scheduling priority to this UE in the time domain, i.e. scheduling delay or buffering delay will be set to a predefined small value (e.g. 20ms in the proposed scheduler) and/or higher scheduling priority to this UE in the frequency Domain. For example, if several users have the same C/I metric in a certain sub-band, the user with a higher QoS requirement will be assigned a preferred sub-band with the highest priority.

4.3 Proposed Scheduler

Based on the above optimization principles, the time and frequency domain scheduler can be optimized by using the UE specific QoS requirements as follows:

1. Time domain scheduler

If a given UE is more sensitive to packet losses, this user will be scheduled with a smaller weight on the buffering delay (e.g. 20ms buffering delay in the proposed scheduler).

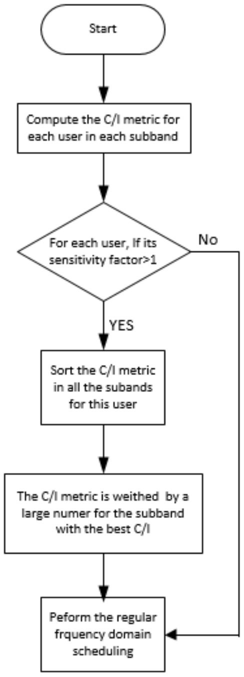

Start
Compute the C/I metric for each user in each subband
For each user, If its sensitivity factor>1
No
YES
Sort the C/I metric in all the subands for this user
The C/I metric is weithed by a large numer for the subband with the best C/I
Peform the regular frquency domain scheduling

Fig. 3. Workflow of the optimized frequency domain scheduler

2. Frequency domain scheduler

If a given UE is more sensitive to packet losses, the C/I metric for the best sub-band of this user will be weighted in the proposed scheduler so that the user can be allocated this best sub-band with much higher priority. The UE specific weighted metric scheduling can also be easily extended to other baseline schedulers. Fig.3 shows the workflow of the optimized frequency domain scheduler. From Fig.3, we can also find that the extra operations resulting from the user specific QoS awareness scheduling are only the weighting operation and sorting operation in all the sub-bands for each packet losses sensitive user. Therefore, the extra complexity is low.

5 User-Specific Frequency Sensitivity QoS Study

Another very promising area of research is a user-specific frequency sensitivity QoS study. For humans, the audible range of frequencies is usually between 20 Hz and 20 kHz. However, there is considerable variation between individuals - especially at the high frequency end, which is primarily affected by a gradual decline with age. Elderly people are normally less sensitive to high frequencies, while younger people are more sensitive to higher frequencies. Fig. 4 shows the hearing loss as a function of the frequency and age[3]. This difference in the sensitivity to higher frequencies can be utilized to further increase system capacity. A frequency sensitivity factor β is defined as the ratio of highest sensitive frequency of a given user to the standard sampling rate 8 KHz. If a user has a frequency sensitivity factor less than 1, the sampling rate can be reduced to 8β KHz, the data rate will be reduced, then the system capacity (i.e. number of concurrent users) will be increased correspondingly. An OPNET experiment was performed to determine the capacity improvement.

6 System Simulation Setup

6.1 System Simulation Configuration

The system simulation was run using the OPNET 17.5 Modeler[15] with the LTE modules. The system simulation configuration is partly based upon LTE macro-cell system simulation baseline parameters [16] as shown in Table 1. In this paper, one single cell with 24 AMR VoIP users was tested for a downlink cross layer scheduler, with an ideal uplink receiver.

6.2 System Simulation Scenarios

Two scenarios were designed and simulated as described in Table 2. In Scenario 1 users have different packet loss sensitivity factors affecting voice quality, while in Scenario 2 in Table 2 user have different frequency sensitivity factors affecting voice quality.

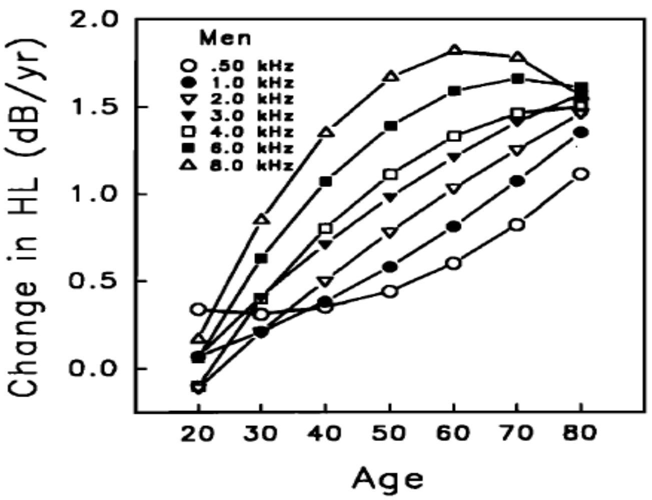


Fig. 4. Hearing loss [HL] as a function of frequency and age [3]

7 Simulation Results

The simulation results for Scenario 1 are shown in Fig. 5-6. The average MOS of all 24 UEs are plotted in Fig. 5. From the figure, we see that greater MOS improvement can be achieved for UEs with larger sensitivity factors and relatively poor MOS (e.g. around 39% MOS improvement for UE24). A view of the MOS of UE24 changing with the time was plotted in Fig.6. It can be seen that the proposed scheduler greatly improves the MOS for UEs that are more sensitive to packet losses and have a relatively poor MOS at the same time. Since UEs with poor MOS need to improve their MOS, the user specific QoS-aware scheduler is very effective in improving the MOS to the desired level.

Table 1. System Simulation Configuration

Parameter	Assumption
Cellular Layout	1 Cell
Cell Radius	1Kilometer
Path loss model	3GPP suburban Macrocell
Mobility model	Random Way Point (RWP) with speed of 0.1km/h
Carrier Frequency	Uplink:1920MHz Downlink:2110MHz
System Bandwidth	5MHz
Channel model	ITU Pedestrian A
Total BS TX power	40dBm
UE power class	23dBm
VoIP codec modes	AMR12.2,AMR10.2K, and mixed codec modes
Number of Users	24 VoIP Users
Scheduler	Dynamic scheduling The proposed scheduler and LTE baseline scheduler
Other assumptions	Ideal uplink receiver(no block error and packet loss) , PDCP compression disabled

Table 2. System Simulation Scenarios

Scenarios	Assumption	
Scenario 1	24 AMR10.2K VoIP Users, each user randomly takes a value for the sensitivity factor α, where $\alpha \in \{0.8, 0.9, 1, 1.1, 1.2\}$, the proposed scheduler and LTE baseline scheduler. Sensitivity factor α is taken in the test as follows:	
	UE index	Sensitivity Factor α
	1-2	0.8
	3	0.9
	4-6	1
	7	1.1
	8-11	1.2
	12-14	0.9
	15	0.8
	16	1
	17	0.8
	18	0.9
	19	1
	20	1.1
	21	1.2
	22	0.9
	23	0.8
	24	1.2
Scenario 2	24 G.711 VoIP users with a freqeuncy sensitivity factor β = 1 24 quasi-G.711 VoIP users with a freqeuncy sensitivity factor β = 0.75, 0.5, 0.25 respectively	

Fig.7 plots the approximate capacity improvement (i.e. number of supportable users) as a function of frequency sensitivity factor β. In the simulation, a rough mapping from the Physical Downlink Shared Channel (PDSCH) load to the system capacity improvement can be done according to the following formula:

$$\text{Capacity impovement for factor } \beta(\%) = \frac{1/(\text{load for factor } \beta)}{1/(\text{load for factor } \beta = 1)} - 1 \tag{7}$$

From Fig.7, we can see that more than 100% capacity improvement can be achieved with a sensitivity factor β of 0.25, while an increase of around 30% can be achieved with a sensitivity factor β of 0.5 and 0.75.

8 Conclusion and Future Research

In this paper, we introduced the concept of user specific QoS requirements and demonstrated their importance and utility in spectral allocation and improving the perceived quality [MOS] for wireless systems. A user specific QoS MOS formula was defined and a novel user specific QoS aware scheduler with low complexity was described that significantly improves the MOS of VoIP users based on the user-specific

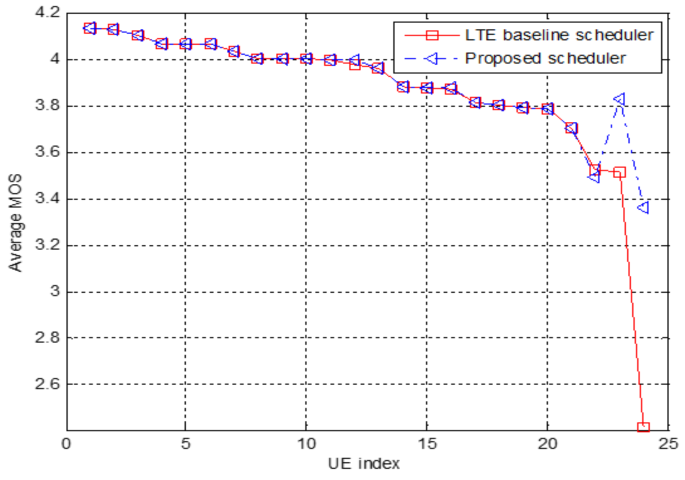


Fig. 5. Average MOS as a function of UE index for AMR10.2K

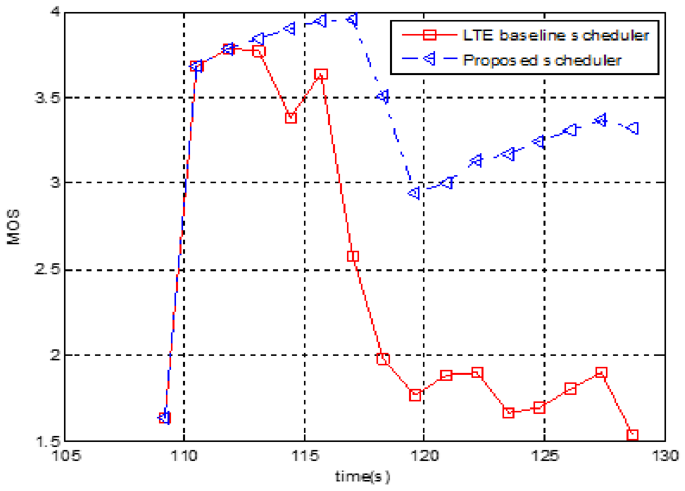


Fig. 6. MOS as a function of time for UE24 for AMR10.2K

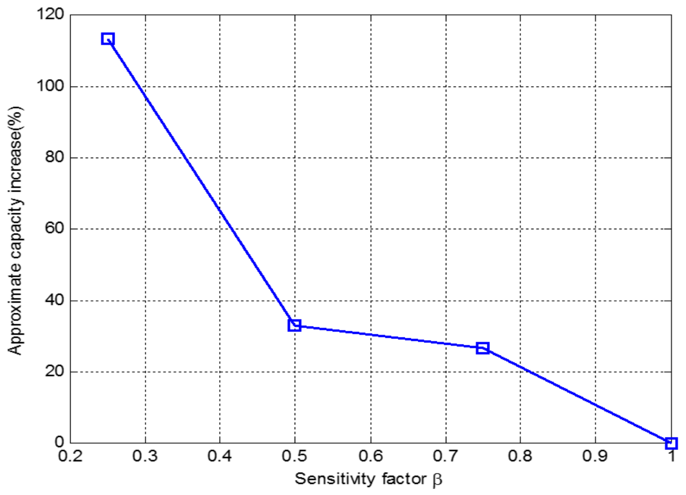


Fig. 7. Approximate capacity improvement as a function of frequency sensitivity factor β

QoS requirements, especially for UEs with relatively poor MOS. The simulation results presented here are only for voice users; however, the same scheduling algorithm will be extended to other applications [e.g., multimedia or data] as one of our future research directions. Moreover, when combined with AMR codecs matched to the different high frequency auditory characteristics of users, higher system capacity as well as comparable MOS levels may be achieved.

Acknowledgment. This research was supported by NSF Grant 1352883. We would like to thanks Drs. Huseyin Arslan, Gabriel Arrobo, and Zygmunt Haas for many stimulating and enlightening discussions during this research.

References

1. Ekstrom, H.: QoS control in the 3GPP evolved packet system. IEEE Commun. Mag. 47, 76–83 (2009)
2. 3GPP: TS 23.203 - V10.9.0 - Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Policy and charging control architecture, TS 123 203 - V10.9.0 - Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Policy and charging control architecture (3GPP TS 23.203 version 10.9.0 Release 10) - ts_123203v100900p.pdf.
3. Pearson, J.D., Morrell, C.H., Gordon-Salant, S., Brant, L.J., Metter, E.J., Klein, L.L., Fozard, J.L.: Gender differences in a longitudinal study of age-associated hearing loss. J. Acoust. Soc. Am. 97, 1196–1205 (1995)
4. Khan, S., Duhovnikov, S., Steinbach, E., Kellerer, W.: MOS-Based Multiuser Multiapplication Cross-Layer Optimization for Mobile Multimedia Communication. Adv. Multimed. 2007, 1–11 (2007)

5. Saul, A., Khan, S., Auer, G., Kellerer, W., Steinbach, E.: Cross-Layer Optimization With Model-Based Parameter Exchange. In: Presented at the Communications, ICC 2007. IEEE International Conference on (2007)
6. Thakolsri, S., Khan, S., Steinbach, E., Kellerer, W.: QoE-Driven Cross-Layer Optimization for High Speed Downlink Packet Access. J. Commun. 4 (2009)
7. Saul, A., Auer, G.: Multiuser Resource Allocation Maximizing the Perceived Quality. EURASIP J. Wirel. Commun. Netw. 2009, 1–15 (2009)
8. Kela, P., Puttonen, J., Kolehmainen, N., Ristaniemi, T., Henttonen, T., Moisio, M.: Dynamic packet scheduling performance in UTRA Long Term Evolution downlink. Presented at the 3rd International Symposium on Wireless Pervasive Computing, ISWPC 2008 (2008)
9. Beh, K.C., Armour, S., Doufexi, A.: Joint Time-Frequency Domain Proportional Fair Scheduler with HARQ for 3GPP LTE Systems. Presented at the Vehicular Technology Conference, 2008, VTC 2008-Fall, IEEE 68th (2008)
10. Telecommunication standardization sector of ITU: G.107: The E-model: a computational model for use in transmission planning (2011), `http://www.itu.int/rec/T-REC-G.107`
11. Fitzpatrick, J.: An E-Model based adaptation algorithm for AMR voice calls. Wireless Days (WD), 2011 IFIP, pp. 1–6 (2011)
12. Cole, R.G., Rosenbluth, J.H.: Voice over IP Performance Monitoring. SIGCOMM Comput. Commun. Rev. 31, 9–24 (2001)
13. Mertz, F., Vary, P.: Efficient Voice Communication inWireless Packet Networks. In: 2008 ITG Conference on Voice Communication (SprachKommunikation), pp. 1–4 (2008)
14. 3GPP: TS 26.071 - V11.0.0 - Mandatory speech CODEC speech processing functions;AMR speech CODEC; General description, `http://www.3gpp.org/ftp/specs/archive/26_series/26.071/26071-b00.zip`
15. OPNET, `http://www.opnet.com/`
16. 3GPP: TS 25.814 - V7.1.0 - Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA), `http://www.3gpp.org/ftp/Specs/archive/25_series/25.814/25814-710.zip`

A Distributed Storage Model for Sensor Networks

Lee Luan Ling

School of Electrical and Computer Engineering, State University of Campinas, Brazil
lee@decom.fee.unicamp.br

Abstract. In this paper we present a novel efficient, simple, local, memoryless and deterministic storage aggregation model for large wireless sensor networks (WSNs). Our goal is to maximize the system storage utilization. While normally most of WSN systems have a vast amount of global storage reserves, their storage capacity may not be fully explored due to the fact that each sensor individually has a very limited low storage capacity. We suggest an aggregated storage model that overcomes this drawback of low individual storage capacity. Our model constructs "on-demand" *distributed storage chains* (DSCs) which search for available un-occupied storage space inside WSNs. Those chains are constructed via deterministic geographic walks; however, their behavior resembles random walks. Simulation has revealed that the proposed storage distribution model is capable of maximizing utilization of WSN storage capacity as well as maintaining network loads geographically uniformly distributed.

Keywords: Distributed algorithms, Expanders, Geo-Routing, P2P, Random walk, Voronoi Diagrams.

1 Introduction

Sensor networks (also SensorNets) are distributed sensing systems comprised of large numbers of autonomous devices that record activities on different regions [1,2,3]. These devices record detailed information about their surrounding environments in order to study or measure some physical and field phenomena. As the number of transistors on a cost-effective chip doubles every year or two [4] (Moore's law), SensorNet technology has become feasible economically which enables new paradigms for network computing. In fact SensorNets have provided the state of the art solution for monitoring extensive observations, for example seismographic activity, weather, surveillance, radioactivity, habitat monitoring, etc.

While the amount of storage capacity for each individual sensor is very limited, the network system as a whole globally possesses a vast amount of aggregated storage space [4]. In terms of its storage capacity the network system seems to be robust, but its information management maneuverability is highly constrained by the low storage resource limitation of each individual sensor. Clearly substituting new sensors with higher storage capacity and/or predicting the amount of monitored activities on each region many not always be feasible; in consequence, considerable research work has been conducted in intelligent exploitation of the overall SensorNet resource capacities. Although there has been a lot of prior work on resource aggregation, most approaches are based on the characteristics of the aggregated resource in order to implement an appropriate aggregation solution. In other words, most solutions are case or scenario dependent.

Another important issue of SensorNets is regarding energy consumption which is widely studied. More specifically the investigations have focused on SensorNet energy optimization. The power consumption in SensorNets can be viewed as a local management problem (within individual sensor – e.g. when it is active) or a global optimization problem within the entire network system. A typical example for the latter issue is the problem of energy holes [5] dealing with the problem of high energy consumption of those sensors that have to process heavy traffic load. Those sensors having depleted their available energy become "energy holes" and, as a consequence, create communication coverage holes which can jeopardize the full functionality and applicability of any SensorNet. The study reported in [5] is based on the "many to one" communication strategy, i.e., all sensors communicate with a central station (sink station). Accordingly, sensors that are closer to a sink point may observe and have to process heavy traffic load towards the sink point also and therefore with high risk of having their energy depleted off very early. Moreover, since the position of a sink point is usually pre-defined, in this work we do not investigate the situation of random energy hole formation. In fact, energy holes can be avoided by making them geographically distant from sink points.

Naor et al [6] present a novel approach for constructing a P2P (Peer to Peer) network topology based on a Distributed Hash Table. The proposed model divides the continuous 1-D interval $I=[0, 1)$ into various non-overlapping subintervals, each one representing a process job. Then, through a continuous-discrete mapping, the resulted network system is a discretization of the continuous graph by division of the underlying space into a cell domain. A similar approach is also used for constructing a GHT (Geographic Hash Table), that is, segmenting a continuous geographic address space in order to associate with discrete home-node set (sensor locations). Ratnasamy et al [11] suggest a *data aggregation* (data centric) model dealing with both communication and storage capacities. All data information related with the same event type (e.g., elephant sighting) is directed to or stored at a common node. Therefore, queries for any specific information should be sent directly to the corresponding hosting sensor without necessity of flooding the system. Under this model, sensors communicate among themselves in order to deliver acquired event information to their hashed location. Unlike storage aggregation, data gathering is based on the type of activity data being stored.

In this paper we address the problem of overloaded activities of sensors where sensors having their storage capacity already depleted (*exhausted sensors*) by their own generated information data. A SensorNet can maximize its storage capacity utilization if events occur at locations which are geographically uniformly distributed. If not, some coverage holes may sooner appear due to those sensors with exhausted storage space. Mostly important, under this circumstance the overall storage may not be fully utilized; therefore, the system may still hold unused storage capacity. If this is the case, the following crucial problem still needs to be solved: how to explore and access this unused storage space. In this work we design a protocol that establishes paths between exhausted sensors and simultaneously accesses sensors with available storage reserves.

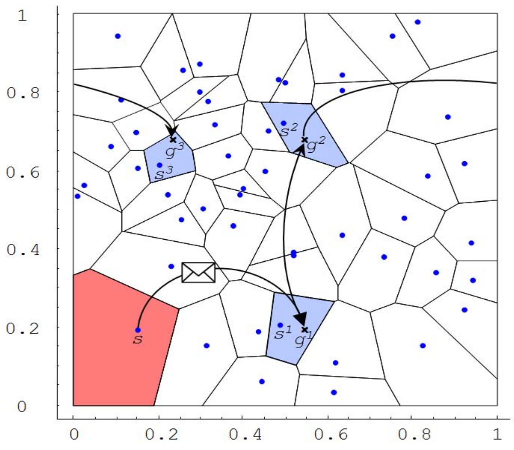


Fig. 1. A Voronoi diagram for a set of nodes; the continuous GG walk starting at *s*, $\omega(s)=(s,g^1,g^2,g^3)$ and its DSC (s,s^1,s^2,s^3)

2 Network Model and Layers

2.1 The Network Model

Without losing generality, we model our wireless sensor network as a set S of n sensors $(s_1, s_2, \ldots, s_n)$ distributed inside the unit square $[0,1]^2$. Let $\{x(s_i), y(s_i)\}$ denote the coordinates of sensor s_i, and assume that the sensors are aware of their own geographical locations. We model the communication network as a unit disk graph [8], on which two sensors can communicate wirelessly if and only if their Euclidian distance is less than the *transmission range* r, which is a parameter of the network. We further assume that the network is connected, i.e., r is large enough to guarantee that there are no isolated components inside a circle of radius r. Whenever an information generator sensor $s_i \in S$ becomes exhausted, it sends and stores the unstored data at available sensor $l_1(s_i) \in S$. If sensor $l_1(s_i)$ is also exhausted, it seeks and forwards information to another available sensor $l_2(s_i) \in S$. This procedure is repeated until the data information is adequately stored. The chain of accessed sensors, originated from the source sensor s_i, is denoted by $\omega(s_i)=(s_i,l_1(s_i),l_2(s_i),\ldots,l_n(s_i))$ and called Distributed Storage Chain (DSC). Our routing protocol, therefore, is composed of a set of instructions for a source sensor to establish the corresponding DSC.

For better description and understanding of our protocol we use a well-known concept of computational geometry called the *Voronoi diagram* [10]. A Voronoi diagram for a set of nodes S in a two dimensional space is a procedure partitioning the input space into non-overlapping Voronoi cells or regions denoted by $Vor(s)$, $s \in S$. A Voronoi cell is a convex region such that all the points inside $Vor(s)$ are closer to s than to any other node of S. In our case, we only consider points in $[0, 1)^2$. We use the Euclidean distance as our distance measure. An example of a Voronoi Diagram for a set of nodes is illustrated in Fig. 1.

2.2 Network Layer - Geographical Routing

Similarly to the GHT [7], our routing protocol is built atop GPSR (Greedy Perimeter Stateless Routing) [11], which provides routing instructions based only on the geographical locations of sensors in the field. This kind of routing holds local, memory-less and unicast characteristics that fit the needs of DSC. GPSR uses two routing modes: *Greedy Forwarding* and *Perimeter Forwarding*. In the Greedy Forwarding mode, packets are routed progressively closer to their destinations. In other words, when a sensor receives a packet, it forwards the packet to its neighbor that is closest to the packet's final destination. The Greedy Forwarding mode cannot be applied whenever the packet reaches a node whose neighbors cannot forward it further to any sensor closer to the destination (known as a void or local minima). In such a case, we change to the perimeter forwarding mode which execute a *planar sub-graph* procedure of the network, i.e., the packet will be routed along the faces of the planar graph using the *right-hand rule*. As soon as the packet being able to reach a sensor that is closer to the destination (closer than the sensor that initiated the perimeter forwarding mode), GPSR returns to execute the greedy forwarding mode.

As we will show later, the continuous nature of DSC requires that messages can be routed not only between two sensors but also between any two geographic-locations. To do so, we use the same solution proved by Ratnasamy et. al. [11]; that is, when a message is addressed to some destination that does not corresponding to a sensor location, we say that the message was routed to the region covered by a particular sensor. If a packet has not reached its destination, it should be related to a sensor s that represents the region covered by this sensor. This representing sensor is called the *Home node* which is the closets sensor to all destinations delimited by the Voronoi cell defined by this Home node.

2.3 The Gabber-Galil Overlay Network

Conceptually a DSC builds dynamic overlay networks which are *"on-demand"* defined by a set of exhausted sensors. Those dynamic networks grow by time (starting as an empty graph). Chain links are dynamically added whenever a sensor or its related chain becomes exhausted. Our overlay network is formed based on the *explicit expander construction* proposed by Gabber and Galil [12]. We show below that the resulted network is a highly connected graph with a linear number of edges.

We formally introduce the continuous Gabber-Galil expander graph: Let $G(V, \varepsilon)$ be a continuous graph within the unit square where $\{x, y\} \in V = [0, 1)^2$ is a continuous set of vertices (points) [9]. $\varepsilon \subset V \times V$ denotes the set of edges in G. We use the Gabber-Galil *transform* [12] (actually only a part of it) to relate each point in V with two Gabber-Galil links. With respect to the link directions, we name them East Link and North Link.

Definition 1. *The Gabber-Galil transform for a point $\{x, y\}$ in $[0, 1)^2$ establishes the following two links:*

$$GG(x,y) = \begin{cases} East:\{x+y, y\} & \mod(1) \\ North:\{x, x+y\} & \mod(1) \end{cases} \tag{1}$$

An edge *({x,y}, {x', y'})* is in *ε iff {x',y'}* matches either the east or north link of *{x, y}*. The major motivation of using this expander over the geographic field is due to a desired expansion property of the continuous Gabber-Galil graph as described established by Theorem 1,

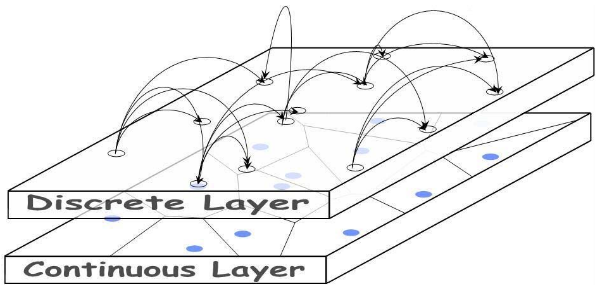


Fig. 2. Two dimensional continuous discrete approach

Theorem 1[12]. *For every set A of points in V such that μ(A) ≤ 0.5, where μ(A) represent the area size covered by A, the following the property holds:*

$$\mu((GG_{East}(A) \bigcup GG_{North}(A))/A) \geq \frac{2-\sqrt{3}}{2}\mu(A) \tag{2}$$

According to Theorem 1, the Gaber-Galil transform enables rapid area expansion. The covered area size expansion rate is superior to 13.3% ($\frac{2-\sqrt{3}}{2} = 0.133$) of the original covered area size only by the first iteration. As a consequence, a few number of iterations of the Gaber-Galil transform of any sensor set size will allow quick coverage of a large network area and therefore localize potential network storage reserves.

Since the output of the Gabber-Galil transform can result in a pair of real numbers, the results of this transformation will indicate the cell regions delimited by their corresponding sensors. Notice that two cell regions are connected if they share adjacent points provided by the real number graph. In our case, the sensors divide the whole area into disjoint Voronoi cells (i.e., regions) and we say that each sensor is a *home node* for all the points inside its Voronoi region. Notice that Voroni cells defined by sensors can be viewed as a continuous-discrete state mapping [6] which relates the real number sets of vertices and links *V* and *ε* with the discrete set of vertices *S* (sensors).

Definition 2. *For a point p ∈ [0, 1)^2 and a set of sensors S located in [0, 1)^2 , let D(p) denotes the Home node(sensor) of p. There is only one sensor s∈S for which p∈Vor(s).*

Now, we can formally define our overlay network as a discrete graph *G(V, E)*. Let *V* = *S* denote a set of sensors that are located on the unit square and let *E* represents a discrete set of edges. *G* describes the overlay network resulting from our protocol (section 3). Definition 3 (and Figure 2) establishes the relation between the set of edges *E* of the discrete graph and the set of edges ε of the continuous graph:

Definition 3. *By the continuous-discrete mapping approach: An* ***undirected edge*** *(s_i,s_j) is in E if and only if:*

$$\exists p, p' \ \ s.t. \ \ (p,p') \in \varepsilon \wedge D(p) = s_i \wedge D(p') = s_j \tag{3}$$

In other words, an edge in E is directed from a point in the region of s_i to a point in the region of s_j.

2.4 System Lifetime

If the network system lifetime is defined as the moment on which at the first time a sensor becomes depleted and fails to save or forward generated information, we can say that the main goal of this work is to prolong the system's lifetime by avoiding the loss of information. Our approach proposed in this work is to design a transmission mechanism that enables exhausted sensors, even having their storage capacity depleted, forward information data to other un-exhausted sensors.

3 Protocol Description

The proposed routing protocol can be described by a simple abstract instruction that runs at each sensor: *If possible store locally else store forward.* In other words, our protocol uses the *store-forward* instruction to build a DSC. The links connecting two nodes (sensors) are established by a deterministic search function to locate available sensors. Since our search function operates deterministically, it can be applied for both tracking available sensors and retrieving the distributed data from destination sensors. An information source should keep the address of the last link (EOC - End of Chain) in order to identify the sensor at which the information data is stored. When an EOC sensor becomes exhausts, a new link is added to the DSC by the routing algorithm, updating the current EOC sensor.

3.1 Distributed Storage Chains

In literature applying several random walks simultaneously to build a graph expander also has demonstrated good efficiency [17]. Notice that the Gabber-Galil expander or transform is a deterministic expander; therefore generating deterministic graphs. It is worth mentioning that data retrieval can be easily achieved by running this Gabber-Galil deterministic mapping. Now we formally define the previously introduced continuous space chain and its corresponding distributed storage chain version.

Definition 4. *A* ***continuous Gabber-Galil (GG) walk*** *of length $k+1$ initiated by site $s=g^0$ is the path $\omega(s) = (g^0,g^1,g^2,...,g^k)$ where $g^j=\{x_j,y_j\}$ and for $1\leq j \leq k$, $(g^{j-1},g^j)\in\varepsilon$. We say that $\omega(s)$ is a* ***chain*** *on the continuous layer.*

A discrete form of ω(s) is a chain denoted as $W(s)=(s, D(g^1), D(g^2),...,D(g^k))$ and can be considered as a special case of the discrete Gabber-Galil walk.

Definition 5. *A distributed storage chain (**DSC**) starting at s, is the discrete form of the continuous chain, where for every step j, $1 \leq j \leq n$, $g^j = GG^{j-1}(g^{j-1})$:*

$$GG^j(x,y) = \begin{cases} GG_{East}(x,y) & if\ j = 0 \bmod(2) \\ GG_{North}(x,y) & if\ j = 1\ \bmod(2) \end{cases} \tag{4}$$

In other words, the continuous chain is alternatively constructed by taking the *East* and *North* links from the Gabber-Galil transform. Note that for a sensor *s*, its continuous chain $\omega(s)=(g_0(s), g_1(s), g_2(s),...)$ can be computed recursively, i.e., $g^{j+1}(s)=GG^j(g^j(s))$. Thus, a sensor can compute locally the $j^{'th}$ location of its continuous space chain.

Fig. 1 illustrates a DSC of length 4 starting at the generator sensor *s*. The continuous space walk is denoted by $\omega(s)=(s, g^1, g^2, g^3)$ where $g^1=GG^0(s)$, and $g^2=GG^1(g^1)=GG^1(GG^0(s))$, etc... The discrete space chain of sensors for this walk is represented by $(s, s^1, .s^2, s^3)$ where $D(g^i)=s^i$.

3.2 Protocol Specifications

Our data storage protocol consists of three basic messages: PUT, GET and ACK. The messages contain the following basic variable parameter: ***pos*** - destination address, (on continuous space layer); notice that it does not have to be the address of any specific sensor; ***gen*** **-** the location of the generator sensor; ***step*** - the current length of the DSC originated from gen; ***data*** **-** the data to be stored; ***user*** - location of the user that issued the query; ***Q*** **-** query, (each sensor is either able to answer *Q* or determine whether such data does not exist in its local memory), ***EOC*** - location of the last link of DSC (continuous address).

```
Algorithm 1. PUT(pos,gen,step,data)
    if FreeMemory(data) then
        StoreLocally(data)
        send ACK(gen,step)
    else send PUT($GG^{step}$(pos), gen,step+1,data)
```

Whenever an exhausted sensor is requested to store data, it issues a PUT packet to its EOC (i.e., pos=EOC) so as to forward that data. A sensor with free memory space, after receiving this data packet, will store this data on behalf of the generator. Then, an ACK packet will be created and sent back by the last sensor to the source generator sensor in order to track and update the current DSC information. Since the Gabber-Galil transform is a deterministic mapping function, any network can easily trace any destination location where the data were stored.

```
Algorithm 2. ACK (gen,step)
        update chainLength=step
        $EOC=g^{step}(s)$
```

Algorithm ACK defines the procedure of chain updating when a ACK message is received by sensor *s*. Basically an ACK packet is sent to the generator so that the generator can update its operational parameters: the current length of the chain and the EOC. Recall that the EOC is the continuous space address of the last element in the DSC (i.e., *step(s)*) which can be determined progressively. Therefore, when a new data is sensed and stored, the generator is quickly informed by a ACK message about the final destination address indicated by the EOC.

```
Algorithm 3. GET (pos, user, Q, EOC, step)
        if DataExist(Q) then send data to user
        else if (pos == EOC) then send "NOT EXIST " to user
    else send GET(GGstep(pos), user, Q, EOC, step + 1)
```

A GET protocol packet has the goal of retrieving stored data information at destination sensors. For this end, the user invokes a GET packet at the source sensor *s* (i.e., *pos* = *s*). If the data is not stored at *s*, the proper source sensor *s* will issue another GET packet that travels along its DSC up to EOC. This session can end with one of two possible situations. If the data is positively found at a sensor along the chain, an reply is sent back to the source sensor and the data is recovered; otherwise, a ”NOT EXIST” message is transmitted back to the source sensor.

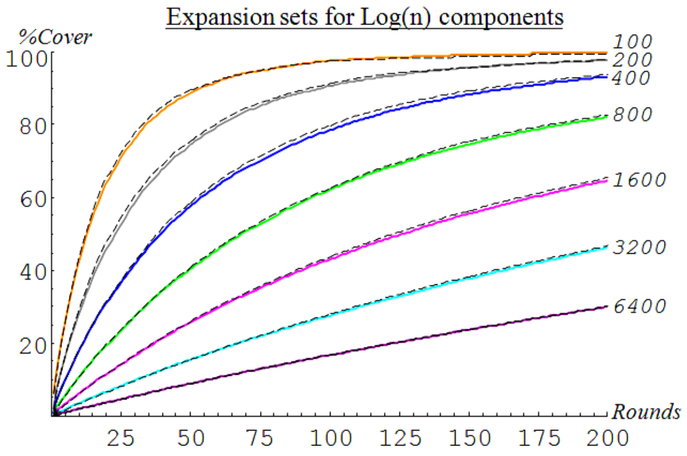


Fig. 3. Simulation results of expansion sets: percentage of coverage (100 to 6400 sensors)

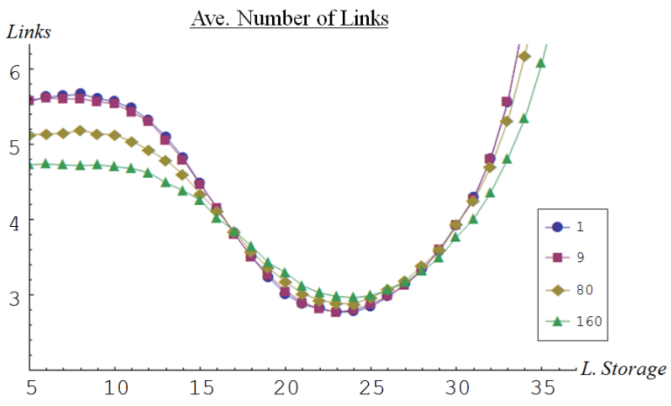


Fig. 4. Average number of links for the STORE sessions

4 Simulation Results

The experimental investigation described in this section intends to evaluate the quality of the DSC algorithm in terms of the efficiency of the proposed data storage and retrieval procedures. For this end, we setup a simulation environment using *Wolfram Mathematica 7*. The designed software program executes some computational geometry procedures required by our protocol (i.e., determining Home nodes on a continuous domain space and implementing multi-domain trees). In this simulation task, the network topologies are established by random deployment of *n* sensors uniformly distributed on a unit square. Eight different network sizes were evaluated, composing of {50, 100, 200, 400, 800, 1600, 3200, 6400} sensors. For each network size we have tested 120 different sensor distribution topologies randomly generated.

4.1 Experiment 1: DSC Expansion Sets

Although a DSC mechanism allows maximum use of network storage capacity by making exhausted sensors forward data information to other still available sensors, the operational cost for additional activities of storage and retrieval operations must be considered. This is due to the fact that the DSC protocol causes an exponential increase of number of links as well as the number of packets traveling inside the network and visiting a large number of exhausted sensors. Ideally it is desirable to reach available sensors as early as possible, possibly avoiding *non-active* links (a link leading to an exhausted sensor). In this experiment, we simulated several deterministic searching strategies over the network with the aim of evaluating the *expansion* quality of these expansion mechanisms and the overhead costs (*non-active* links) caused by running the DSC algorithm. Actually the proposed and adopted Gabber-Galil transform has demonstrated its simplicity and efficiency in search tasks through North and East links alternatively.

To determine the expansion properties of DSC, we performed the following simulation. We examined five different primary sensor component sets with size $k = \{1, log_2(n), 5\%, 10\%, 20\%\}$ representing hot-spots (network regions presenting high sensor monitoring activities). A primary component set $S_0 \subset S$ whose elements s^i's were randomly selected ($s^i \in S_0$, $0 \leq i < k$) from $S=\{s_1, s_2, s_3, ..., s_n\}$. Each site $s^i \in S_0$ individually initiated its own DSC procedure. At each iteration step, denoted by variable j, $0 \leq j < 200$, we let S_j be the set of all sensors that have already been visited (or covered) by at least one of the k chains. The same simulation procedure is repeated 200 times with different initial S_0 primary component sets randomly selected. Our analysis focused on the evaluation of the performance of DSC. Notice that for each searching step performed on the overlay network, considerable GPSR hops were generated. Moreover, we assume each packet has a constant-size overhead regardless of network system size and topology.

Fig. 3 presents the growing rate of $|S_j|$ (expressing in percentage (%) of the total number of sensors) in function of step number j for network topologies varying from 100 to 6400 sensors with the primary components set size fixed at $(log_2(n))$. To evaluate the obtained simulation results, we compare our DSC deterministic walk with a random walk, denoted by ***rXY***. ***rXY*** randomly chooses a Home node from the unit square at each searching step. It is interesting to observe that the sensor selection by *rXY* does not conform with uniform probability distribution. In fact the sensor selection probabolity is proportional to the *Vor(s)* area of sensor *s*. This problem of different Voronoi cell sizes was also discussed in [2]. The area coverage rate provided by *rXY* can be considered as the best among all the expansion procedures reported in the literature and becomes the reference of the comparison carried out here. For our experimental comparison, the area coverage rate provided by *rXY* is represented by the dashed curves in Fig. 3.

According to Fig. 3, clearly the performances of the DSC deterministic routing algorithm are very similar to those provided by *rXY* almost independent of network size. These results prove that the Gabber-Galil transform is a good deterministic search function in which, mostly importante, each sensor relies only on its local information. Although the results presented here are only for $k = log_2(n)$, our experiment revealed that the walk behavior of our DCS is close to that of *rXY* under any network configuration and size (even for a single hot-spot, $k = 1$).

4.2 Experiment 2: Data Storage and Retrieval

In this part of experimental investigation, we evaluated the efficiency of the DSC algorithms in terms of activities of information generation, storage and retrieval performed by network sensors as proposed in Section 3.2. We set out to evaluate two basic metrics, STORE and RETRIEVE. STORE refers to the number of messages that the algorithm needs to generate and send through the overlay network (using PUT message). RETRIEVE is the number of hops a data packet has to travel in a data retrieval operation (using a GET message).

To test the proposed protocol, we considered and evaluate network environments under the so-called extreme hot-spots conditions (high event activity rates occurring at few number of specific sensors). We dedicated our simulation on a network with 1600 sensors. Each sensor has a local memory size equal to $M=100$ data units. During the simulation, we generated event *activity* (data units) up to 80% of the total system storage capacity in the following way: Let $S_{HS}=\{s^1,s^2,s^3,...,s^k\}$ be a set of k hot-spot sensors. With probability $p_{HS}=0.8$, an event activity is generated by a sensor s_i chosen randomly from the set S_{HS} , and with probability $1-p_{HS}=0.2$ the activity being generated by another sensor also randomly selected from S/S_{HS} (the complement sensor set of S_{HS}.) Therefore, it is almost certain that each sensor $s \in S_{HS}$ needs to construct its own DSC in order to accommodate all the activity data it detected. This is due to the fact that those hot spot sensors usually will sooner become exhausted.

In order to prevent endless data forwarding in the network, we fix TTL (Time to Live) at 16 hops for PUT packets. Interesting enough, our simulation revealed that only less than 0.05% of messages were discarded after expiration of TTL.

We also investigated situations under an additional restriction: local capacity storage for each sensor (i.e., the amount of memory allocated to local activities) varying between 0-40 units. In addition, since a sensor generator receives ACK messages for successive storage operations, it can also localize non- active links. Therefore, on a retrieval operation, it is possible that a user avoid sending messages to those links. Fig. 4 shows the simulation results for 1600-sensor network and local storage varying 0-40% with 4 different hot-spot sets. The results for each local storage size represent the expected value among repetitions with 100 different random hot-spot sets of the same size. Fig. 4 and 5 present show the average chain lengths (i.e., number of hops) for STORE and RETRIEVE for different sizes of local storage and different amounts of generated activities (in % of the total system storage capacity). Fig. 5 in fact represents the worst case in terms of information retrieval cost because it provides the chain length, or the cost needed to retrieve data from the last sensor of the chain.

Fig. 4 and 5 provide important information about network behaviors under the effects imposed by the proposed protocol. For low activity (up to 50% of the total storage), we report that the cost difference between STORAGE and RETRIVE is minimum, because localizing available sensors for storage is straightforward. On the other hand, as the majority of sensors in the network have become exhausted soon, it is more difficult to track available sensors immediately. Therefore STORAGE operation will become more expensive with the increase of activity rate (e.g. 80%, see Figures 4 and 5). Here, we consider the cost difference between STORE and RETRIEVE in terms of the amount of energy that was wasted due to non-active link sessions on which the GG transform was applied to reach exhausted sensors. Generally the operational cost of Gabber-Galil transform becomes more expensive with the increase of activities. The STORE measure grows exponentially fast and the RETRIEVE fortunately shows more moderate growing rate (to local storage of 35 data units for this given simulation). Moreover, according to Fig. 4 and 5, there are minimum cost points for STORE and RETRIVE operations, respectively, what suggests that the local storage size can be a network parameter for network performance optimization, i.e., minimum chain length.

Fig. 6 shows the Voronoi diagram of a 1600 sensors network. Each cell is illustrated by a distinct brightness level representing the average number of times it was selected but unable to store data locally (becaming exhausted). Clearly there is some correlation between the size of the Voronoi cells and the number of storage failures. Notice also the larger the cell the brighter it appears. We conclude that the Gabber-Galil transform can adequately cover the network field uniformly regardless of site locations. Therefore, if sensors are geographically uniformly distributed or the Voroni cells are of equal size, better performance of SensorNets is expected to achieve.

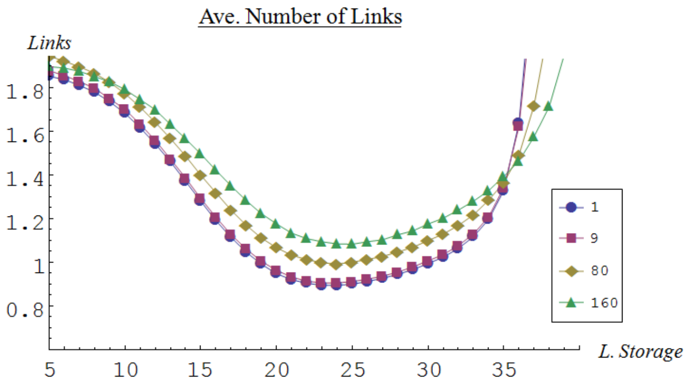


Fig. 5. Average number of links for RETRIEVE sessions

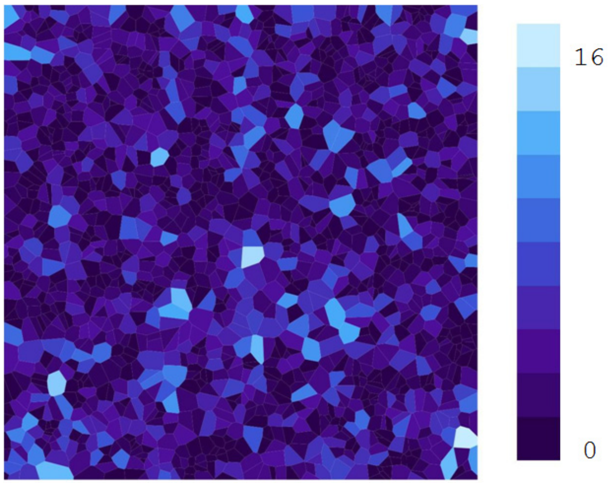


Fig. 6. Load balancing analysis for a 1600 WSN: number of visits on each sensor region

5 Conclusion

In this work, we present a novel new information storage strategy for WSN. First we provided some background about expanders and introduced the original Gabber-Galil expander graphs for a discrete grid network. Then we developed our continuous expander graphs about the deployment of sensor in field and defined a 2-D continuous discrete approach that relates sensor nodes with the proposed continuous expander graphs. Our main contribution is this work consists of implementing distributed storage chains (DSCs) via a deterministic walk (but with random walk behavior) for information storage and distribution based on geographic expander graphs, GPSR and Veroni diagrams. Simulation results showed that the proposed DSC mechanism allows sensor networks to operate satisfactorily even under extreme conditions (e.g., information not geographically uniformly generated). Since the proposed DSC protocol is deterministic, there is no need to maintain any routing tables or information about network structure or topology. The simulation revealed that the DSC implemented in SensorNet has similar behavior of multiple random walks [13] over expander graphs and, mostly important, is highly efficient in searching available storage reserves. For future work, we will extend and deepen our investigation into the following issues: comparison with performance of random walk [7], energy issues on WSN [15], lifetime of WSN [16] and topology discovery algorithms for WSN [14].

References

1. Estrin, D., Govindan, R., Hiedemann, J.: Next century challenges: Scalable coor-dination in sensor network. USC/ Information Science Institute (1999)
2. Lewis, F.L.: Wireless sensor networks. In: Smart Environments, pp. 11-46. John Wiley&Sons (2005)
3. Dimakis, A.G., Sarwate, A.D., Wainwright, M.J.: Geographic gossip: efficient aggregation for sensor networks. In: The Fifth International Conference on In-formation Processing in Sensor Networks, pp. 69–76. ACM (2006)
4. Culler, D., Estrin, D., Srivastava, M.: Guest editors' introduction: Overview of Sensor Networks. IEEE Computer 37(8), 41–49 (2004)
5. Wu, X., Chen, G., Das, S.K.: Avoiding energy holes in wireless sensor networks with non-uniform node distribution. IEEE Transactions on Parallel and Distributed Systems 19, 710–720 (2008)
6. Naor, M., Wieder, U.: Novel Architectures for P2P Applications: The continuous-discrete approach. ACM Trans. Algorithms 3 (2007)
7. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless net-works. In: The 6th Annual International Conference on Mobile Computing and Networking, pp. 243–254. ACM, New York (2000)
8. Clark, B., Colbourn, C., Johnson, D.: Unit disk graphs. Discrete Math. 86, 165–177 (1990)
9. Diestel, R.: Graph theory. Springer (2006)
10. Berg, M.: Computational Geometry: algorithms and applications. Springer (2000)
11. Rantnasamy, S., Shenker, S., Govinadan, R.: Data centric storage in SensorNets with GHT. A Geographic Hash Table, Mobile Networks and Application 8(4), 427–442 (2003)
12. Gabber, O., Galil, Z.: Explicit constructions of linear size super-concentrators. In: The 20th Annual Symposium on Foundations of Computer Science, pp. 364–370. IEEE Computer Society, Washington, DC (1979)
13. Alon, N., Avin, C., Kouck´y, M., Kozma, G., Lotker, Z.: M. Tuttle, R.: Many random walks are faster than one. In: The 20th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 119–128, New York (2008)

14. Gkantsidis, C., Mihail, M., Saberi, A.: Random Walks in Peer-to-Peer Networks: algorithms and evaluation. Performance Evaluation 63, 241–263 (2006)
15. Shiue, H.Y., Yu, G.J., Sheu, J.P.: Energy Hole Healing Protocol for Surveillance Sensor Networks. In: Workshop on Wireless, Ad Hoc, and Sensor Networks (WASN 2005), Taoyuan, Taiwan (2005)
16. Dietrich, I., Dressler, F.: On the lifetime of wireless sensor networks. ACM Transactions on Sensor Networks 5(1), 1–39 (2009)
17. Clark, B., Colbourn, C., Johnson, D.: Unit disk graphs. Discrete Math. 86, 165–177 (1990)

Relation between Irregular Sampling and Estimated Covariance for Closed-Loop Tracking Method

Bei-bei Miao and Xue-bo Jin

School of Computer and Information Engineering,
Beijing Technology and Business University, Beijing, 100048, China
Miaobeibei1@163.com, xuebojin@gmail.com

Abstract. Regular sampling methods have a widely use in the target trajectory tracking fields and the tracking results are accurate but not fast enough sometimes especially with the long-data measurement. Irregular sampling methods for target tracking can trace the target with less time cost but the result may not very accurate due to the reduced information. This paper aims to find a balance between the computing speed and estimation performance. Based on an irregular sampling closed-loop tracking method, a sample with 2991 points simulated for 2D tracking. We conclude that our method can get a good estimation performance with high computing speed when the Irregular Sampling Rate is 66.1%.

Keywords: closed-loop tracking, irregular sampling rate, Kalman filter, estimation performance, estimation covariance.

1 Introduction

In many applications, there are so many measurements needed to track on real-time. Select some of the measurements to estimate the real trajectory can save computing cost, but will lead to irregular sampling tracking [1]. In this case, the classical models [2] and the estimation methods are no longer applicable, because they are all based on the regular sampling, i.e., visual tracking [3] and the tracking with autonomous underwater vehicles and lagrangian drifters [4].

The paper [5] provides a solution for the target tracking system with randomly sampling measurement, where the irregular sampling interval is transformed to a time-varying parameter by calculating the matrix exponential, and the dynamic parameter is estimated by the online estimated state with Yule-Walker method. This closed-loop estimation method can obtain fast tracking for the long trajectory.

In order to implement the fast video tracking, a method to track target with irregular sampling is developed in [6], which developed an adaptive system dynamic model under irregular sampling and gave the tracking algorithm based on Kalman filter. The model concluded that the irregularity of sampling time has no effect on the estimation performance when with the same number of measurements.

The paper [5] discussed the convergence of the method and paper [6] provided in detail and established adaptive model for indoor moving targets. The paper concluded that regular sampling and irregular sampling can get the same estimation variance if the measurement range is the same for a period of time. The simulate results also show that the reduction of sampling points results in a reduction of the measurement information, and causes the performance degrade. Thus we found the relationship between the sampling points and the performance of system is very important, and it is necessary to find a balance between the computing speed and the estimating performance.

This paper studies the estimation of performance about the time and the number of random sampling points, by which found a balance between the computing speed and estimation performance. The paper is organized as follows: Section 2 gives the estimation method based on Kalman filter. The simulations and experiments are provided in section 3. Finally, concluding remarks are given in Section 4.

2 Tracking Based on Adaptive Statistics Model

Let x , $\dot{\mathrm{x}}$ and $\ddot{\mathrm{x}}$ be the target position, velocity, and acceleration along a generic direction, respectively. Specifically, $\ddot{\mathrm{x}}(t) = a(t)$. In this section, the state vector is always taken to be $x = [\mathrm{x}, \dot{\mathrm{x}}, \ddot{\mathrm{x}}]'$ along the generic direction, unless stated otherwise explicitly.

We get the discrete-time equivalent as the following

$$x(k+1) = \Phi(k+1 \mid k)x(k) + U(k)\bar{a} + w(k) \tag{1}$$

$$\Phi(k+1,k) = \begin{bmatrix} 1 & T & (\alpha T - 1 + e^{-\alpha T})/\alpha^2 \\ 0 & 1 & (1-e^{-\alpha T})/\alpha \\ 0 & 0 & e^{-\alpha T} \end{bmatrix} \tag{2}$$

$$U(k) = \begin{bmatrix} T^2/2 \\ T \\ 1 \end{bmatrix} - \begin{bmatrix} (\alpha T - 1 + e^{-\alpha T})/\alpha^2 \\ (1-e^{-\alpha T})/\alpha \\ e^{-\alpha T} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha}(-T + \frac{\alpha T^2}{2} + \frac{1-e^{-\alpha T}}{\alpha}) \\ T - \frac{1-e^{-\alpha T}}{\alpha} \\ 1-e^{-\alpha T} \end{bmatrix} \tag{3}$$

where $x(k) = \left[\mathrm{x}(k) \quad \dot{\mathrm{x}}(k) \quad \ddot{\mathrm{x}}(k)\right]'$, $\bar{a}$ is the mean of acceleration, the parameter $\alpha = 1/\tau$ is the reciprocal of the maneuver time constant τ and thus α depends on how long the maneuver lasts. T is the sampling interval.

The covariance of the $w(k)$ as

$$Q(k) = E[w(k)w'(k)] = \delta_w^2 \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \tag{4}$$

where

$$\begin{aligned}
&q_{11}=\frac{1}{2\alpha^5}\left[1-e^{-2\alpha T}+2\alpha T+\frac{2\alpha^3T^3}{3}-2\alpha^2T^2-4\alpha Te^{-\alpha T}\right] && q_{12}=\frac{1}{2\alpha^4}\left[e^{-2\alpha T}+1-2e^{-\alpha T}+2\alpha Te^{-\alpha T}-2\alpha T+\alpha^2T^2\right]\\
&q_{13}=\frac{1}{2\alpha^3}\left[1-e^{-2\alpha T}-2\alpha Te^{-\alpha T}\right] && q_{22}=\frac{1}{2\alpha^3}\left[4e^{-\alpha T}-3-e^{-2\alpha T}+2\alpha T\right]\\
&q_{23}=\frac{1}{2\alpha^2}\left[e^{-2\alpha T}+1-2\alpha T\right] && q_{33}=\frac{1}{2\alpha}\left[1-e^{-2\alpha T}\right]
\end{aligned} \tag{5}$$

Let the parameter $\delta_a^2=E\left[a(t)^2\right]$ be the "instantaneous variance" of the acceleration, which has the conversion relationship with δ_w^2 as $\delta_w^2=2\alpha\delta_a^2$. We have the discrete time equivalent of the acceleration as

$$a(k+1)=\beta a(k)+w^a(k) \tag{6}$$

where $w^a(k)$ is a zero-mean white noise sequence with variance

$$\delta_{aw}^2=\delta_a^2(1-\beta^2) \qquad \beta=e^{-\alpha T} \tag{7}$$

For a first-order stationary Markov process, we have the statistics relation between the autocorrelation function with α and δ_{aw}^2 as $\begin{bmatrix} r(0) & r(1)\\ r(1) & r(0)\end{bmatrix}\begin{bmatrix}1\\ -\beta\end{bmatrix}=\begin{bmatrix}\delta_{aw}^2\\ 0\end{bmatrix}$, where $r(0)$ and $r(1)$ are autocorrelation of the acceleration, that means

$$\begin{aligned}
r_N(0)&=\frac{1}{N}\sum_{k=0}^{N-1}\hat{a}(k)\hat{a}(k)\\
r_N(1)&=\frac{1}{N}\sum_{k=1}^{N-1}\hat{a}(k)\hat{a}(k-1)
\end{aligned} \tag{8}$$

and

$$\beta=\frac{r(1)}{r(0)} \qquad \delta_{aw}^2=r(0)-\alpha r(1) \tag{9}$$

Then we can use Eq. (7) to get α and δ_a^2 by β and δ_{aw}^2.

The discrete state–space model of the tracking system is as the following

$$\begin{aligned}
x(k+1)&=\Phi(k+1\,|\,k)x(k)+U(k)\overline{a}+w(k)\\
y(k)&=H(k)x(k)+v(k)
\end{aligned} \tag{10}$$

where $x=[\mathrm{x},\dot{\mathrm{x}},\ddot{\mathrm{x}}]'$ is the state of the system to be estimated and whose initial mean and covariance are known as X_0 and P_0, $w(k)$ and $v(k)$ are white noise with zero mean and independent of the initial state x_0, $y(k)$ is the measurement vector, $H(k)$ is measurement matrices and $v(k)$ is measurement noise with known variance R.

The following is the algorithm in the predictor–corrector form by Kalman filter, which is convenient for implementation:

Initialization: $k=0$

$$\hat{x}(0\,|\,0)=x_0 \quad P(0\,|\,0)=P_0 \quad \alpha(0)=\alpha_0 \quad \delta_w^2(0)=\delta_{w0}^2 \quad r_0(0)=\ddot{\mathrm{x}}_0\cdot\ddot{\mathrm{x}}_0 \quad r_0(1)=\ddot{\mathrm{x}}_0$$

Recursion: $k:=k+1$

a) Prediction:

$$\hat{x}(k+1\,|\,k)=\Phi(k+1,k)\hat{x}(k\,|\,k)+U(k)\overline{a} \tag{11}$$

$$P(k+1\,|\,k)=\Phi(k+1,k)P(k\,|\,k)\Phi'(k+1,k)+Q(k) \tag{12}$$

b) State update:

$$\hat{x}(k+1\,|\,k+1)=\hat{x}(k+1\,|\,k)+K(k+1)[y(K+1)-H(k+1)\hat{x}(k+1\,|\,k)] \tag{13}$$

$$K(k+1) = P(k+1|k)H'(k+1)[H(k+1)P(k+1|k)H'(k+1)+R]' \tag{14}$$

$$P(k+1|k+1) = [I-K(k+1)H(k+1)]P(k+1|k) \tag{15}$$

c) Adaptation: Set the estimation of acceleration $\hat{a}(k)$ as

$$\hat{a}(k) = \ddot{\mathrm{x}}(k|k) \tag{16}$$

d) Parameter update:

$\hat{a}(k)$ is the estimation of acceleration, we have

$$\bar{a} = \frac{1}{N}\sum_{k=0}^{N-1}\hat{a}(k) \tag{17}$$

when $K \le k_0$, $\alpha = 1/20$, $\delta_{\mathrm{a}}^2 = 25\times(4-\pi)/\pi$, then use Eq. (2) - (4) to get the system parameter $\Phi(k+1,k)$, $U(k)$ and $Q(k)$.

when $K > k_0$, the following is used to update the system matrix.

$$r_k(1) = r_{k-1}(1) + \frac{1}{k}\left[\hat{a}(k)\hat{a}(k-1) - r_{k-1}(1)\right] \tag{18}$$

$$r_k(0) = r_{k-1}(0) + \frac{1}{k}\left[\hat{a}(k)\hat{a}(k) - r_{k-1}(0)\right] \tag{19}$$

$$\beta = \frac{r(1)}{r(0)} \qquad \delta_{aw}^2 = r(0) - \alpha r(1) \tag{20}$$

$$\delta_a^2 = \delta_{aw}^2 / (1-\beta^2) \tag{21}$$

$$\alpha = -\frac{\ln\beta}{T} \tag{22}$$

then use Eq. (19) - (22) to get the system parameter $\Phi(k+1,k)$, $U(k)$ and $Q(k)$.

Note1: $r_k(0)$ and $r_k(1)$ are the autocorrelation functions, which need to have statistical data. In the simulation, we set the two parameters change after 4-20 steps. Here we set $k_0 = 4$.

The conditions above are not enough to make it possible that the target tracking is achieved when the sampling ratio changed. To ensure the convergence of the algorithm is the key to guarantee the performance of algorithm

Based on the closed-loop estimation algorithm [6], we can see the parameter used to estimate state is an estimated one and similarly, the estimated states to calculate the parameters α_i and $\delta_{a\,i}^2$ have estimation errors, too. So it's important to guarantee the convergence of the estimation of the states and parameters. From paper [5], we can see if one step predictive covariance is bounded, i.e., $P(k|k-1) | \le P_0$, $P(k+1|k)$ must be bounded with the fact $\hat{Q}_d(k) | \le Q_0$. And $K(k+1)$ must be a bounded matrix and $\hat{x}(k+1|k+1)$ must be bounded, too.

Note3: we set δ_a^2 a bounded value 10000 to avoid system divergence.

Note4: We update the model parameters α and δ_a when $\beta > 0$ only, because if $\beta < 0$, the system will divergence.

3 Simulation Results

Based on the algorithm developed here we will analyze the relation between the random sampling interval, the location and number of sampling points, i.e., Irregular Sampling Rate (ISR) and the tracking performance

$$ISR = \frac{\text{the number of sampling points}}{\text{the total number of the sample}} \tag{19}$$

We can see lower ISR means larger irregular sampling interval.

The proposed algorithm is applied to a two dimensional planar tracking problem to verify the performance through a series of simulation runs. The measurement noises are generated as white Gaussian random numbers with variance *R*. We assume *R* is a given parameter, as $R = 25$ .The initial state estimate x_0 and covariance P_0 are assumed to be $x_0 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]'$ and $P_0 = diag(10,10,10,10,10,10)$. The actual trajectory of the target (contents 2991 points) is shown in Fig.1. Due to the measurement noise we can only get the trajectory sequence with noise shown in Fig.2. The solid line is the actual trajectory and the star line represents the locations of sampling points.

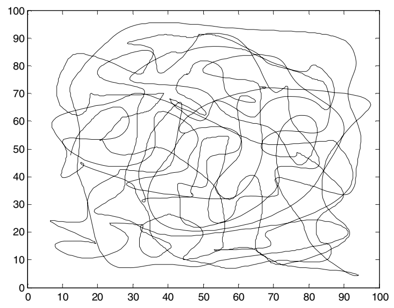


Fig. 1. The actual trajectory of the target

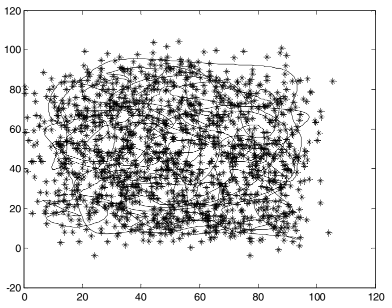


Fig. 2. The measurement of the target

Based on the develop algorithm about irregular sampling interval for closed-loop tracking, we give nine types of estimation results about different irregular random sampling rate under a sample with 2991 points, that is 90.3%,79.8%,70.4%, 60%, 49.8%,40.5%,28.5%,21.5%,9.5%.

To better illustrate our results, we give the contradistinction of the real and the estimates trajectories shown in Fig.3. The estimations of horizontal and longitudinal axis show in Fig.4 and Fig.5 respectively. The location estimation errors show in Fig.8 and Fig.9 respectively.

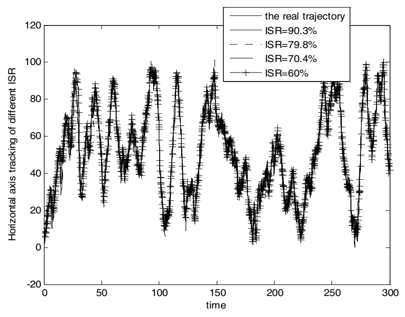


Fig. 3. The real and the estimations of horizontal axis with ISR as 90.3%, 79.8%, 70.4% and 60%

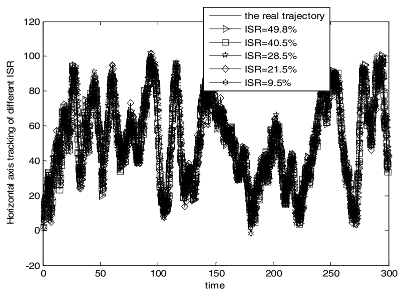


Fig. 4. The real and the estimations of horizontal axis with ISR as 49.8%%, 40.5%, 28.5% , 21.5and 9.5%

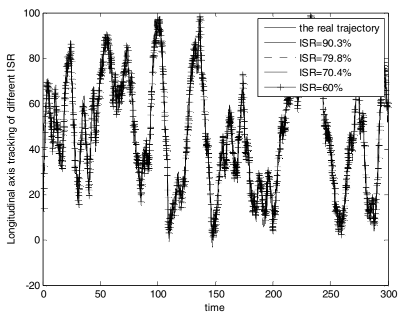


Fig. 5. The real and the estimations of longitudinal axis with ISR as 90.3%, 79.8%, 70.4% and 60%

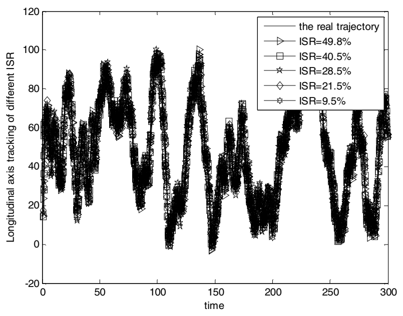


Fig. 6. The real and the estimations of longitudinal axis with ISR as 49.8%%, 40.5%, 28.5% ,21.5and 9.5%

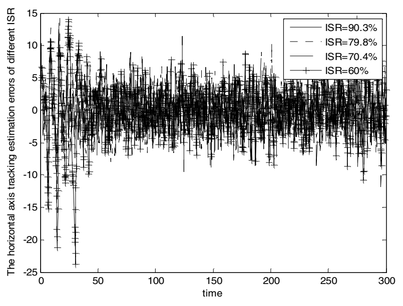


Fig. 7. The estimation errors of horizontal axis with ISR as 90.3%, 79.8%, 70.4% and 60%

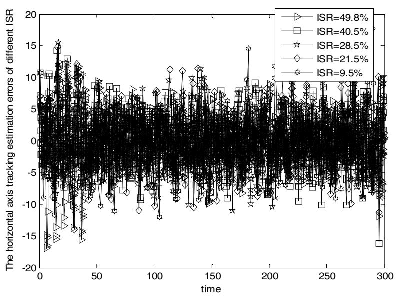


Fig. 8. The estimation errors of horizontal axis with ISR as 49.8%%, 40.5%, 28.5%, 21.5 and 9.5%

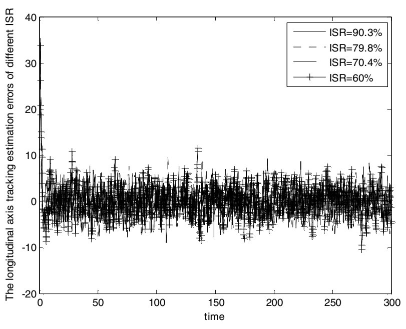


Fig. 9. The estimation errors of horizontal axis with ISR as 90.3%, 79.8%, 70.4% and 60%

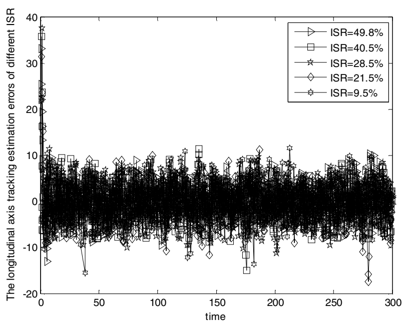


Fig. 10. The estimation errors of horizontal axis with ISR as 49.8%%, 40.5%, 28.5%, 21.5 and 9.5%

To illustrate how the irregular sampling rate affects estimation performance, we choose the estimated covariance as a metric.

$$\begin{aligned}
&\hat{x}(i) - x(i) = e_x(i) \\
&\hat{y}(i) - y(i) = e_y(i) \\
&COV = \sum_{i=1}^{N} e^2(i) \Big/ N \\
&e^2(i) = e^2{}_x(i) + e^2{}_y(i)
\end{aligned} \tag{20}$$

Table 1 gives some of the estimated covariance under different irregular sampling rate (ISR). Fig.11 gives the covariance of the location of sampling points of different ISR.

Table 1. Relation between Irregular Sampling Rate and Estimated Covariance

ISR	90. 3%	79. 8%	70. 4%	60%	49. 8%	40. 5%	28. 5%	21. 5%	9. 5%
Covariance Of X axis	9. 874	9. 929	10. 527	17. 424	16. 381	13. 224	14. 730	19. 657	22. 217
Covariance Of Y axis	7. 883	9. 210	9. 340	10. 732	13. 562	13. 002	19. 010	17. 392	23. 263

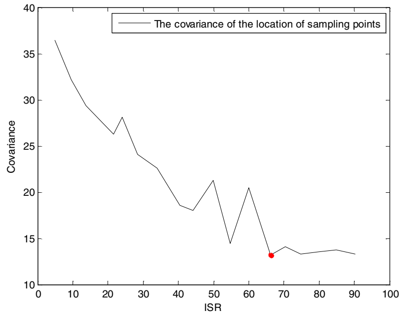


Fig. 11. The covariance of the location of sampling points

We can see the sampling rate doesn't have much impact on the estimated variance. The lower sampling rate means the less measured data is get, and the less useful information can be provided, therefore the estimate is faster. From fig.11 we can conclude that there exist balance between the computing speed and the estimating performance when ISR=66.1%.

4 Conclusions

The main contribution of this paper is to analysis how the randomly selected measurements and irregular sampling rate will effects the estimation performance and cost time on closed-loop tracking. We found it can reduce the computing cost greatly. But we can also see with the larger ISR, the estimation performance will decline because less useful information is used for tracking. Thus we conclude that a balance between the computing speed and the estimating performance is existed.

Acknowledgments. This work is partially supported by NSFC under Grant No. 61273002, 60971119 and the Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions No. CIT&TCD201304025.

References

1. Jian, W.: Underwater multi-target tracking theory, pp. 52–64. Northwestern University Press (2009)
2. You, H., Jianjuan, X., Jingwei, Z.: Radar Data Processing and Applications, pp. 178–201. Electronic Industry Press, Beijing (2011)
3. Huaping, L., Fuchun, S., Yuanlong, Y.: Multitask Extreme Learning Machine for Visual Tracking. Science and Business Media, pp. 1–14. Springer, New York (2014)
4. Jnaneshwar, D., Frédéric, P., Thom, M., O'Reilly, T., Monique, M., John, R., Kanna, R., Gaurav, S.S.: Simultaneous tracking and sampling of dynamic oceanographic features with autonomous underwater vehicles and lagrangian drifters. Tracts in Advanced Robotic 79, 27–30 (2014)
5. Xue-Bo, J., Jingjing, D., Jia, B.: Fast tracking for video target tracking. Applied Mechanics and Materials. Sensors, Measurement and Intelligent Materials, 303-306 (2013)
6. Xue-bo, J., Xiaofeng, L., Tingli, S., Yan, S., Beibei, M.: Closed-Loop Estimation for Randomly Sampled Measurements in Target Tracking System. Mathematical Problems in Engineering, Article ID 315908 (2014)

Author Index