Ming Xu
Yinwei Zhan
Jiannong Cao
Yijun Liu (Eds.)

Advanced Parallel Processing Technologies

LNCS 4847

7th International Symposium, APPT 2007
Guangzhou, China, November 2007
Proceedings

Springer

Lecture Notes in Computer Science 4847

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Ming Xu Yinwei Zhan Jiannong Cao
Yijun Liu (Eds.)

Advanced Parallel Processing Technologies

7th International Symposium, APPT 2007
Guangzhou, China, November 22-23, 2007
Proceedings

Springer

Volume Editors

Ming Xu
National University of Defense Technology
Computer School
Changsha, Hunan 410073, China
E-mail: xuming-64@hotmail.com

Yinwei Zhan
Yijun Liu
Guangdong University of Technology
Faculty of Computer Science
Guangzhou, Guandong 510090, China
E-mail: {ywzhan, yjliu}@gdut.edu.cn

Jiannong Cao
The Hong Kong Polytechnic University
Department of Computing
Hung Hom, Kowloon, Hong Kong, China
E-mail: csjcao@comp.polyu.edu.hk

Library of Congress Control Number: 2007939056

CR Subject Classification (1998): D, B, C, F.1-3, G.1-2

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

ISSN 0302-9743
ISBN-10 3-540-76836-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-76836-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12193387 06/3180 5 4 3 2 1 0

Preface

We are currently witnessing a proliferation in parallel and distributed processing technologies and applications. However, more new technologies have ushered in unprecented challenges to the research community across the range of high-performance computing, multi-core microprocessor architecture, networks and pervasive computing, as well as new paradigm computing issues.

APPT 2007 was sponsored by the China Computer Federation, in cooperation with TCPP of the Institute for Electrical and Electronics Engineers (IEEE). The highly positive responses to the previous APPT workshops encouraged us to continue this international event. This year, APPT was upgraded to the International Symposium on Advanced Parallel Processing Technologies. However, it kept its traditional flavor by sharing of the underlying theories and applications, and the establishment of new and long-term collaborative channels. And it will continue to provide a forum for researchers, professionals, and industrial practitioners from around the world to report on new advances in high-performance architecture and software, as well as to identify issues and directions for research and development in the new era of evolving technologies.

The success of APPT 2007 was a result of the hard work and planning of a large group of renowned researchers from around the world, who served on the Technical Program Committee and the Organizing Committee. Their invaluable efforts in developing this technical program are most gratefully acknowledged. In particular, we would like to thank the Program Co-chairs, Xin Chen, Xuejun Yang, and Albert Y. Zomaya.

We also would like to take this opportunity to thank our keynote speakers: Arndt Bode from the Technical University of Munich and Barbara Chapman from Houston University. The symposium also invited David W. Yen from Sun Microsystems Inc. to offer an interesting talk on multi-core microprocessor product. Their views on different aspects of the challenges we all face were of high value.

Last but not least, the conference would not have been possible without the generous support of our industrial sponsor, Sun Microsystems Inc. We hope you find the papers to be both stimulating and enjoyable!

November 2007

Xingming Zhou
Arndt Bode

Message from the Program Committee Co-chairs

Since 1995, the Advanced Parallel Processing Technologies (APPT) workshop series has provided a forum for international, cross-disciplinary parallel and distributed processing technologies. APPT 2007 was the seventh event in the series.

It was our pleasure to hold this symposium (originally workshop) in Guangzhou. During the conference, participants had the opportunity to obtain the latest information on a variety of aspects of parallel and distributed processing theories, applications, and practices.

This year, we received 346 full manuscripts from researchers and practitioners of 12 countries and regions. Each paper was peer-reviewed so that most papers had at least two anonymous referees. Papers were reviewed and selected based on their originality, significance, correctness, relevance, and clarity of presentation. Only 78 papers were accepted for presentation at the symposium, representing an acceptance rate of 22.5%. Overall, the program struck a comfortable balance between applied and theoretically oriented papers. All accepted papers are included in the proceedings. We appreciate Springer for accepting to publish the proceedings again as part of the LNCS series.

We would like to acknowledge the support of the Computing College, Guangdong University of Technology for taking care of every fine detail in the operation of the symposium. In particular, we mention Yinwei Zhan, the local Organizing Chair, and Yijun Liu, Zhenkun Li, Xiufen Fu for their notable endeavors to make this conference successful. We also express our deepest gratitude to the Computer Architecture Professional Committee members, who offered us valuable advice and suggestions. Without their devotion and contribution, we could not have had a fruitful conference.

On behalf of the Program Committee, we would like to express our heartfelt thanks to everyone who attended APPT 2007!

Xin Chen
Xuejun Yang
Albert Y. Zomaya

Organization

General Co-chairs

Xingming Zhou, National Laboratory for Parallel and Distributed Processing, China
A. Bode, Technical University of Munich, Germany

Program Co-chairs

Xin Chen, Guangdong University of Technology, China
Xuejun Yang, National Laboratory for Parallel and Distributed Processing, China
Albert Y. Zomaya, University of Sydney, Australia

Program Committee Members

Binxing Fang, Harbin Institute of Technology, China
Xinda Lu, Shanghai Jiao Tong University, China
Weimin Zheng, Tsinghua University, China
Xinsong Liu, Electronical Sciences University, China
Siwei Luo, Beijing Jiaotong University, China
Song Shen, Institute No. 706, Aeronautic Industry Inc., China
Jiannong Cao, Hong Kong Polytechnic University, China
Xiangdong Hu, Jiangnan Computing Institute, China
Xiaodong Wang, National Laboratory for Parallel and Distributed Processing, China
Zhiwei Xu, Chinese Academy of Science, China
Zhenzhou Ji, Harbin Institute of Technology, China
Xiaoming Li, Peking University, China
Dongsheng Wang, Tsinghua University, China
Cheng-Zhong Xu, Wayne State University, USA
Wentong Cai, Nanyang Technological University, Singapore
Rodrigo de Mello, University of Sao Paulo, Brazil
Srinivas Aluru, Iowa State University, USA
John Feo, Cray Inc., USA
Kurt Rothermel, University of Stuttgart, Germany
Laurence T. Yang, St. Francis Xavier University, Canada
Eric Aubanel, University of New Brunswick, Canada
Jacques Bahi, University of Franche-Comté, France
Subhash Bhalla, University of Aizu, Japan
Jie Wu, Florida Atlantic University, USA
Jingling Xue, University of New South Wales, Australia
Zahari Zlatev, National Environmental Research Institute, Denmark
Jemal H. Abawajy, Deakin University, Australia

Jie Li, University of Tsukuba, Japan
Martin Buecker, Aachen University of Technology, Germany
Beniamino Di Martino, Second University of Naples, Italy
Andrei Doncescu, University of West French Indies, France
George A. Gravvanis, Democritus University of Thrace, Greece
Minyi Guo, University of Aizu, Japan
Weijia Jia, City University of Hong Kong, China
Helen Karatza, Aristotle University of Thessaloniki, Greece
Ajay Kshemkalyani, University of Illinois, Chicago, USA
Gerhard Joubert, Technische Universität Clausthal, Germany
Thomas Rauber, University of Bayreuth, Germany
Virendra C. Bhavsar, University of New Brunswick, Canada

Publication Chair

Jiannong Cao, Hong Kong PolytechnicUniversity, China

Panel Chair

Qian Zhang, Hong Kong University of Science and Technology, China

Organizing Chair

Yinwei Zhan, Guangdong University of Technology, China
Ming Xu, National University of Defense Technology, China

Demonstration and Exhibit Chair

Yijun Liu, Guangdong University of Technology, China

Industry Liaison

Yong Tong, National Sun Yat-Sen University, China

Publicity Chair

Bingbing Zhou, University of Sydney, Australia

Finance Chair

Zhenkun Li, Guangdong University of Technology, China

Reviewers

Jemal H. Abawajy
Srinivas Aluru
Eric Aubanel
Jacques Bahi
Subhash Bhalla
Virendrakumar C. Bhavsar
Martin Buecker
Wentong Cai
Zhicai Cai
Zhiping Cai
Jiannong Cao
Issac Chan
Wenguang Chen
Beniamino Di Martino
Andrei Doncescu
Xiaoshe Dong
Qiang Dou
Yong Dou
Xiaoya Fan
John Feo
Tony Fong
George Gravvanis
Changguo Guo
Minyi Guo
Xiaoxing Guo
Weihong Han
Fengru He
Hongjun He
An Hong
Fangyong Hou
Chuanhe Huang
Zhenzhou Ji
Xiaohua Jia
Zhiping Jia
Jingfei Jiang
Xiaohong Jiang
Gerhard Joubert
Helen Karatza
Ajay Kshemkalyani
Victor Lee
Hong Li
Mengjun Li
Tiejun Li
XinSong Liu
Yijun Liu
Hongyi Lu
Li Luo
Xinda Luo
Zhigang Luo
Xiaoguang Mao
Xinjun Mao
Rodrigo Mello
Zhiyong Peng
Depei Qian
Zili Shao
Li Shen
Song Shen
Dianxi Shi
Jinshu Su
Caixia Sun
Alfred Tan
Qingping Tan
Yong Tang
Cho-li Wang
Dongsheng Wang
Guojun Wang
Sufeng Wang
Xiaodong Wang
Xingwei Wang
Yijie Wang
Yongwen Wang
Zhijun Wang
Jun Xia
Bin Xiao
Canwen Xiao
Jitian Xiao
Nong Xiao
Xiaoqiang Xiao
Chang sheng Xie
Cheng-Zhong Xu
Jingling Xue
Ming Xu
Laurence T. Yang
Danlin Yao
Jianping Yin
Wanrong Yu
Binyu Zang
Yinwei Zhan
Gongxuan Zhang
Heying Zhang
Yuelong Zhao
Wenhua Zeng
Weimin Zheng
Yi Zheng
Zahari Zlatev
Chuanqi Zhu
Peidong Zhu
Shurong Zou

Table of Contents

Invited Talks

Session 1 – Advanced Microprocessor Architecture

Session 2 – Parallel Distributed System Architectures

Session 3 – Grid Computing

Session 4 – Interconnection Networks

Session 5 – Network Protocols

Session 6 – Pervasive and Mobile Computing Architectures

Session 7 – Task Scheduling and Load Balancing

Session 8 – Software Engineering

Session 8 – Other Issues

Scalability for Petaflops systems

Arndt Bode

Technical University of Munich, Germany
http://wwwbode.cs.tum.edu/~bode/

Abstract. Future very high end systems, petaflops computers, will be megaprocessors or megacores with a million or more active processors. This can be derived both by extrapolation of the processor number of the leading systems in the TOP500 and by the consideration of multi- and many-core microprocessors for energy efficiency reasons. Part of processors could also be application specific accelerators as latest microprocessor architectures support interfaces to such devices. The large number of processors will also impose fault tolerance strategies making the system architectures highly heterogeneous and dynamic. To sum up: petaflops systems will be massively parallel and use heterogeneous and dynamic processor arrangements. Such architectures pose the question of scalability and programmability in general. The talk describes the challanges of such systems for existing application programs, programming languages and models as well as programming tools.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, p. 1, 2007.

Chip Multi-Threading and the SPARC Evolution

David W. Yen

Executive Vice President, Microelectronics
Sun Microsystems, Inc.

The multi-core, multi-threaded (Chip Multi-Threading, or CMT) CPU is a disruptive technology arrived just in time to further boost computing performance at moderate energy cost. While similar to the SMP servers in the 90's, the richness of threads and the finer granularity of parallelism of CMT CPU-based systems do open up a new programming paradigm. System virtualization on such platforms makes the concept more intuitive and provides a "soft landing".

Sun Microsystems has been working on CMT SPARC processors since 2002. The success of UltraSPARC T1 and T2 processors released in December 2005 and August 2007, respectively, bodes well for the entire CMT SPARC processor roadmap. Sun has further made available to the community at large the OpenSPARC T1 and T2, open sourced versions based on the UltraSPARC T1 and T2 under GPL.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, p. 2, 2007.

The Multicore Programming Challenge

Barbara Chapman

Houston University, USA
http://www2.cs.uh.edu/~chapman/

Abstract. Dual-core machines are now actively marketed for desktop and home computing. Systems with a larger number of cores exist, and more are planned. Some cores are capable of executing multiple threads. At the very high end, programmers need to design codes for execution by thousands of processes or threads and have begun to consider how to write programs that can scale to hundreds of thousands of threads. Clearly, the future is multi- and many-core, as well as many-threaded.

In the past, most application developers could rely on Moore's Law to provide them with steady performance improvements. But we have entered an era in which they may have to expend considerable effort if their codes are to exploit the processing power offered by next-generation platforms. At least in the medium term, a broad variety of parallel applications will need to be developed.

Existing shared memory parallel programming APIs were not necessarily designed for general-purpose computing or with many threads in mind. Distributed memory paradigms do not necessarily allow the expression of fine-grained parallelism or provide full exploitation of architectural features. The fact that threads share some resources in multicore systems makes it hard to reason about the impact of program modifications on performance and results may be surprising. Will programmers be able to use multicore platforms effectively?

In this presentation, we discuss the challenges posed by multicore technology, review recent work on programming languages that is potentially interesting for multicore platforms, and discuss on-going activities to extend compiler technology in ways that may help the multicore programmer.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, p. 3, 2007.

Replication-Based Partial Dynamic Scheduling on Heterogeneous Network Processors

Zhiyong Yu[1], Zhiyi Yang[1], Fan Zhang[1], Zhiwen Yu[2], and Tuanqing Zhang[1]

[1] School of Computer Science, Northwestern Polytechnical University, P.R. China
yuzhiyong@mail.nwpu.edu.cn
[2] Academic Center for Computing and Media Studies, Kyoto University, Japan
yu@ccm.media.kyoto-u.ac.jp

Abstract. It is a great challenge to map network processing tasks to processing resources of advanced network processors, which are heterogeneous and multi-threading multiprocessor System-on-Chip. This paper proposes a novel scheduling algorithm, called Replication-based Partial Dynamic Scheduling (RPDS). It aims to improve the NP performance by combining the strategies of partial dynamic mapping and task replication with a 2-phase scheduling. RPDS differs from existing solutions in several aspects, e.g., the processing elements are heterogeneous, fully-connected, and multi-threading, the application is decomposed into directed acyclic graph tasks with continuous data-packets, and scheduling is conducted at both of initialization and run-time. Experimental results showed our algorithm could increase the largest average throughput by about 30% than those without dynamic phase replication.

Keywords: scheduling, network processors, task replication, partial dynamic scheduling, directed acyclic graph.

1 Introduction

The Internet has evolved from a simple store-and-forward network to a complex communication infrastructure. In order to meet demands on security, flexibility and performance of increasingly complex network services, network traffic not only needs to be forwarded, but processed on network devices such as routers. The programmable on-Chip Multi-Processors (CMP) called network processors (NPs) hence appeared. One of the difficulties of application development on such kind of hardware platform is to handle processing resources scheduling. And it also has some other restricts and requirements such as strong real-time, high throughput, low power, small instruction space, changing traffic load, etc., which make it a great challenge to solve this problem.

Scheduling of processing resources is basically to decide which task should be processed on given processing resource at a given time, to achieve the optimal goal. Within NPs, this problem is concretely the mapping from tasks to processing elements (PEs). The tasks are relatively independent code blocks which are decomposed from network applications by using two main methods, i.e. pipelining and directed acyclic graph (DAG).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 4–13, 2007.

The optimizing problem of mapping tasks to PEs is NP-complete [1]. So the practical goal is to get the approximate optimal result. Manual mapping is ineffective and fallible when the system architecture and application are complex [2], and previous research on automatic mapping did not consider the characteristics of advanced NPs.

According to weakest-link principle, the performance of the whole system relies on a few bottlenecks. So we can improve the system performance by abating them. When a task is identified to be a bottleneck, there are usually two solving methods, one is deepening the pipeline, which can't be changed after software compilation, the other is duplicating the task executable code to let it occupy more processing resources at the time of execution. If the bottleneck of a system is changing, the method of task replication can efficiently track the changes and abate the bottleneck.

Therefore, to map network processing tasks to processing resources of advanced complex network processors, this paper proposes a novel scheduling algorithm, called Replication-based Partial Dynamic Scheduling (RPDS). It combines the strategies of partial dynamic mapping and task replication together in NP scheduling that aims to improve the network processing performance in terms of throughput and delay.

The rest of this paper is organized as follows. In Sect. 2, we describe related work in NP scheduling and highlight the distinctive aspects of our approach. In Sect. 3, the details of problem formalization, processing models, and algorithm procedure are proposed. Section 4 presents the evaluation method, simulation tool and experimental results. Finally, Sect. 5 concludes the paper.

2 Related Work

Previous research of mapping tasks to PEs on NPs mainly utilized linear programming and heuristic algorithms, e.g., list scheduling, randomized mapping, and genetic algorithms. In linear programming method, the mapping problem is transformed to a linear programming problem to handled through greedy heuristic [3] or randomized rounding [4]. The list scheduling sorts all tasks according to their priorities and chooses a PE for each task based on a particular rule. Ramaswamy et al. [5] use "criticality" as task priority. Wolf et al. [6] propose two predictive scheduling algorithms, LAP and EFQ, both of which are in nature based on list scheduling. The basic idea of randomized mapping is to randomly choose a valid mapping and evaluate its performance and repeat this process certain times. [7] and [8] present randomized mapping algorithms with different models for performance evaluation. Genetic algorithm maintains a population of candidate solutions that evolves over time and ultimately converges. Yan et al. [2] generate the initial population by utilizing Monte Carlo method. However, the above algorithms made a lot of assumptions and simplifications:

– **Assumptions of PE architecture.** PEs are supposed to be homogeneous, i.e., execution time of a task processed on different PEs are the same; PEs

are supposed to be linked as pipelining; PEs are supposed not to contain hardware multi-threads. Actually these assumptions are not true in advanced NPs' hardware architecture.
- **Simplification of task partition.** Existing algorithms usually choose pipelining tasks, i.e., except the beginning task and the ending task, each task has and only has one predecessor and one successor. This method can not take full advantage of parallelism of NPs. Describing the network application with DAG is more natural. It reflects characteristics of classification, synchronization, and parallelism of data-packets processing.
- **Simplification of scheduling trigger.** Scheduling occasions can be classified into static scheduling, dynamic scheduling, and partial dynamic scheduling [9]. Partial dynamic scheduling is the trade-off of the former two, in which partial tasks are assigned off-line, and others at run-time. It has low computation cost and can achieve local optimal solution at least.

Our work differs from and perhaps outperforms previous work in several aspects. First, the NP platform is different. We use the advanced hardware architecture, which is heterogeneous, fully-connected, and multi-threading. Second, we adopt partial dynamic mapping, which has been rarely studied in existing NP scheduling. Third, although the strategy of task replication has been deeply studied in cluster systems in the context of scientific computing [10], the task model is very different from ours. Furthermore, we are the first to combine task replication and partial dynamic mapping in NP scheduling.

3 RPDS Algorithm

3.1 Problem Formalization

The scheduling problem is expressed by 5-tuple as following:

$$\Pi = (\boldsymbol{G}, \boldsymbol{D}, \boldsymbol{P}, \boldsymbol{\Theta}, \boldsymbol{\Omega}) \quad . \tag{1}$$

$\boldsymbol{G} = (\boldsymbol{T}, \boldsymbol{E})$ is the dependent relationship graph of tasks, which is usually a DAG. It takes elements in $\boldsymbol{T}$ as nodes, and elements in $\boldsymbol{E}$ as directed edges. $\boldsymbol{T} = \{T_1, T_2, \ldots, T_m\}$ is the set of tasks partitioned from the application. $\langle T_i, T_j \rangle \in \boldsymbol{E}\,(i, j = 1, 2, \ldots, m)$ denotes that T_j is processed after T_i, and there is data transferring from T_i to T_j.

$\boldsymbol{D} = \{D_1, D_2, \ldots, D_l\}$ describes the characteristics of data-packets being processed, such as arrival time and which type of process is needed. $D_i = \langle t_i, G_i \rangle\,(i = 1, 2, \ldots, l)$, t_i is the arrival time of the packet, and G_i is a sub-graph of $\boldsymbol{G}$, i.e., the packet needs to be processed by partial or all of the tasks.

$\boldsymbol{P} = \{P_1, P_2, \ldots, P_n\}$ is the set of PEs in the system. Each PE can load several tasks with each one costs time t_{l}. Each PE has r hardware multi-threads. Every two PEs can communicate directly with the delay of time t_{c}, and communicating delays of inner-PE are ignored.

$\boldsymbol{\Theta}$ is an $m \times n$ matrix. An element of it θ_{ij} denotes the execution time of the task T_i on the PE $P_j\,(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$.

$\boldsymbol{\Omega}$ is an $m \times n$ matrix. An element of it ω_{ij} denotes the number of the task T_i on the PE $P_j (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$.

The scheduling problem of DAG network processing tasks to heterogeneous multiprocessors is, given input $\boldsymbol{G}$, $\boldsymbol{D}$, $\boldsymbol{P}$, $\boldsymbol{\Theta}$, to get output $\boldsymbol{\Omega}$ and achieve the optimal goal of the system.

3.2 Processing Model

We present the abstract models for network device and processing. Two definitions are presented at first:

Definition 1. $D_i.RET$ D_i

Definition 2. $D_i.RCT$ D_i

We assume that the application has been decomposed to $\boldsymbol{T}$ appropriately, and $m < n \times r$, i.e., the number of tasks is less then the total number of threads. One node may have multiple successors, which means the application has conditional branches. For example, the sub-graphs of $\boldsymbol{G}$ are $G_1, G_2, \ldots, G_7$, which represent the processing paths of all types of data-packets (see Fig. 1).

Each thread has its own data buffer. When multiple packets arrive, they are organized as a FIFO queue in the buffer (see Fig. 2).There are four states, unoccupied, blocked, running, and ready, of a thread, which can be transited from one to another in a certain condition.

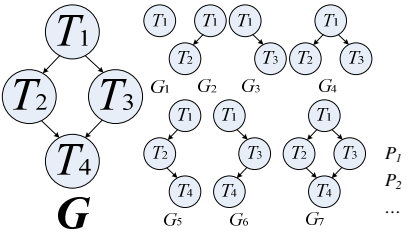


Fig. 1. An example of sub-graphs

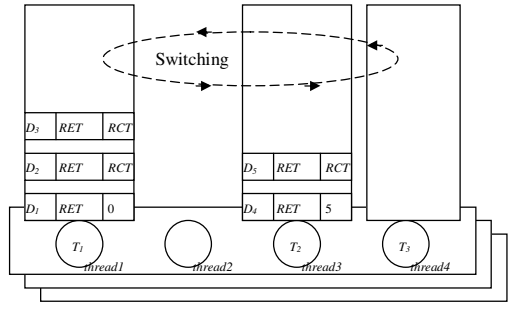


Fig. 2. The architecture of thread buffer

The load state of the PE P_k can be busy, normal, or idle. For a period of detecting time t_{d}, the summation time when P_k is running is t_{e}. Then the utilization rate of this PE is:

$$P_k.UR = \frac{t_{\mathrm{e}}}{t_{\mathrm{d}}} \quad . \tag{2}$$

Given load upper limit λ_1 and load lower limit λ_2, if $P_k.UR \geqslant \lambda_1$, P_k is busy; if $Pk.UR < \lambda_2$, P_k is idle; P_k is normal when $P_k.UR$ falls between λ_1 and λ_2.

For modern processor, communicating is independent with processing. RCT of all data-packets in the buffer minus 1 till 0 after every time unit; RET of the first data-packet in the queue minus 1 till 0 after every time unit if its RCT is 0, while RET of other data-packets remain unchanged; if $D_i.RET$ and $D_i.RCT$ are both 0, the data-packet D_i is finished on this task, and is passed to the tail of successor task queue, resets $D_i.RET$ as θ_{jk} (T_j is the successor task, and P_k is the processor that T_j is loaded), $D_i.RCT$ as t_{C} (different PE) or 0 (the same PE). For a particular task T_j, the buffer of corresponding thread contains data-packets D_1, D_2, ..., D_s. We define $T_j.EEF$ as the earliest expected finish time of T_j:

$$T_j.EEF = \sum_{i=1}^{s} (D_i.RET + D_i.RCT) \quad . \tag{3}$$

The value of EEF implies how busy the task is. For all tasks at the moment, the task whose EEF value is the biggest is the

3.3 Algorithm Procedure

Cost function is used to measure the fitness of mapping results, which is the key of list scheduling algorithm. To take into account execution time, load balance and communication overhead, the cost function is defined as follows:

$$F = a\times\sum_{j=1}^{n}\sum_{i=1}^{m}\omega_{ij}\theta_{ij} + b\times\frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=1}^{m}\omega_{ij}\theta_{ij} - \frac{\sum_{j=1}^{n}\sum_{i=1}^{m}\omega_{ij}\theta_{ij}}{n}\right)^2 + c\times\sum_{j=1}^{m}\sum_{i=1}^{j}\beta_{ij}t_{\mathrm{C}} \quad . \tag{4}$$

where $\beta_{ij} = 1, \text{if}\,(\langle T_i, T_j\rangle \in \boldsymbol{E}\ \&\ \forall k, \omega_{ik}\times\omega_{jk} = 0)$; or 0, else. The every addend respectively means the linear sum of all execution time, the variance of the execution time of every PE, and the linear sum of all communication delay. a, b, c are corresponding weights. The main procedure of RPDS algorithm is described as follows:

- **Static phase scheduling.** At the initialization of NPs, all tasks are organized as an ordering list. First, calculating the difference between the shortest execution time and the hypo-shortest execution time of each task, the larger the difference is, the higher the task priority is. Then for the task with the highest priority in the list, a PE among those contain unoccupied threads is selected to make the value of cost function F minimal, to which the task is allocated. This process is repeated until all tasks are assigned. At the end of this step the result $\boldsymbol{\Omega}_0$ is obtained. As long as the number of tasks m is less than the total number of threads $n \times r$, each task can occupy a corresponding thread. Apparently there are some redundant unoccupied threads after static phase scheduling, which will be fully utilized at the next phase.
- **Dynamic phase scheduling.** During the run-time of NPs, the PEs' states are detected every t_{d} time. If all PEs are busy, it's unable to adjust, and if all PEs are idle, it's unnecessary to adjust. Therefore when there are some

busy PEs and some idle PEs, the bottleneck task found in busy PEs are duplicated. For each idle PE, the value of cost function F is calculated when the bottleneck task replication is loaded on it, and the PE that makes F the minimal is finally chosen. If all threads on the idle PE is occupied, the task whose EEF is 0 is removed. The pseudo code is given in Fig. 3.

Input: $\boldsymbol{G}, \boldsymbol{\Theta}, \boldsymbol{\Omega}_0, time, P_k.UR, T_j.EEF (k = 1, 2, \ldots, n, j = 1, 2, \ldots, m)$
Output: $\boldsymbol{\Omega}_{time}$

```
While t_d|time & ∃P_k.UR ⩾ λ1 & P_k.UR < λ2
    T_bottleneck ← T_j : max{T_j.EEF}
    For each P_k.UR < λ2
        If all threads in P_k are occupied
            If ∃T_j.EEF = 0 ∉ Ω_0 in P_k
                remove T_j
            Else
                continue
        load T_bottleneck
        calculate F_k
    If more than one F_k
        P_chosen ← P_k : min{F_k}
        P_k except P_chosen roll back //remain not changed
    Return Ω_time
End While
```

Fig. 3. The pseudo code of dynamic phase scheduling

4 Performance Evaluation

4.1 Evaluation Metrics

We use average delay and average throughput as metrics to evaluate the algorithm. For each data-packet D_i, its arrival time is $D_i.t_{\text{receive}}$, and its finished time is $D_i.t_{\text{finish}}$, then the delay of this data-packet is $D_i.Delay = D_i.t_{\text{finish}} - D_i.t_{\text{receive}}$. The average delay and throughput of l data-packets is:

$$Average\_Delay = \frac{1}{l}\sum_{i=1}^{l} D_i.Delay \ . \tag{5}$$

$$Average\_Throughput = \frac{l}{D_l.t_{\text{finish}} - D_1.t_{\text{finish}}} \ . \tag{6}$$

These metrics can work only if $\boldsymbol{G}$, $\boldsymbol{D}$, $\boldsymbol{P}$, and $\boldsymbol{\Theta}$ are the same.

4.2 Simulation Tool

We developed a simulation tool called , which implemented the processing model presented in Sect. 3.2. The input was a configuration file in which the

task graph ($\boldsymbol{G}$), packets sending sequence ($\boldsymbol{D}$), PEs ($\boldsymbol{P}$), execution times ($\boldsymbol{\Theta}$), and other parameter values were specified. The outputs included the arrival and finished time of all data-packets (t_{receive}, t_{finish}), the earliest expected finish time of all tasks at each detecting time (EEF), the utilization rates of all PEs at each detecting time (UR), and all mapping results ever have ($\boldsymbol{\Omega}$). Uniform virtual time unit was used in simulation.

Specially, to specify $D_i = \langle ti, Gi \rangle$ for every data-packet separately is time-consuming because there are thousands of packets in the experiment. For t_i, we assumed that the packet sending intervals follow the exponential distribution. For G_i, we added probabilities of data-packets transferred from T_i to T_j. The execution time ranged from 0 to 100, whereas 0 means that the task is not executable on that PE.

4.3 Experimental Results

The choice of parameters is important to the experiments. We used some parameters as default (see Table 1, and varied others to observe their effect to performance.

( ) The DAG in this experiment is presented in Fig. 1, where $m = 4, n = 4, r = 3$. The default branch probability is 1. The execution time of each task is shown in Table 2. To verify the performance of RPDS, we implemented several variations (i.e., different types) of it, which are presented in Table 3.

Table 1. Default parameters

Parameter	t_c	t_l	t_d	λ_1	λ_2	a	b	c
Value	10	20	500	0.9	0.7	0.5	0.1	0.4

Table 2. Execution time

	P_1	P_2	P_3	P_4
T_1	86	57	68	85
T_2	29	61	52	18
T_3	0	53	16	68
T_4	94	6	15	86

Table 3. RPDS algorithm variations

Type ID	Schedule Multi-DAGs at static phase	Have dynamic phase	Allow synonymous tasks in one PE
2	No	Yes	No
4	No	No	No
6	Yes	Yes	No
8	Yes	No	No
10	No	Yes	Yes

Sending data-packets at the constant speed every time unit respectively (e.g., 0.01 denotes that the average sending interval is 100 time units), the results are shown in Fig. 4 and Fig. 5.

We can observe that the average delay and average throughput are both increasing along with the increase of the constant speed. The performance of RPDS variations ranks as: Type-10 > Type-2 > Type-6 > Type-8 > Type-4. Let the delay bound be 1500, we can see that the throughput rate (throughput / sending speed) keeps 1. If exceeding this bound, the throughputs do not increase

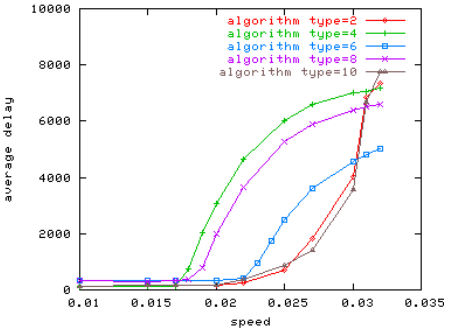


Fig. 4. Packet speed vs. delay

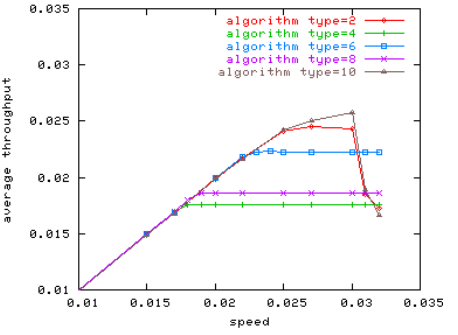


Fig. 5. Packet speed vs. throughput

any more but have trends to decrease. When the packet sending speed is lower than 0.017, the five variations are grouped into two classes: Type-2, Type-4, and Type-10 vs. Type-6 and Type-8. The average delays of the former are about 50% lower than those of the latter. But as the speed increasing, the two classes turn to be: Type-2, Type-6, and Type-10 vs. Type-4 and Type-8. The largest acceptable speeds of the former are 30% larger than those of the latter.

That is to say, when the workload is light, scheduling without static replication is superior to that with static replication; while the workload is heavy, scheduling with dynamic replication performs better than that without dynamic replication. The reason is that there is no need to duplicate at light workload, and furthermore the delay is increased after replication because of the frequent transferring between PEs of data-packets. Dynamic replication abated the pressure of the bottleneck task effectively at heavy workload, balanced the tasks among PEs, and therefore increased the throughput.

( ) In this experiment, the DAG, execution time, and algorithm types are the same as those in experiment (1). The packet sending speed and branch probabilities are different, which are shown in Table 4.

Table 4. Packet sending speed; Branch probabilities of $T_1 \rightarrow T_2/T_1 \rightarrow T_3$

Time	0–10000	10000–20000	20000–30000	30000–40000	40000-50000
Speed	0.017	0.0185	0.0195	0.0235	0.026
Time	0–15000	15000–30000	30000-		
Probability	0.4/0.4	0.1/0.9	0.9/0.1		

We selected Type-10 (with dynamic replication) and Type-8 (without dynamic replication) to compare with each other. The results are shown in Fig. 6 and Fig. 7.

This experiment shows the reason why RPDS can reduce the delay and improve the throughput in more detail. The delay of packets in Type-10 algorithm

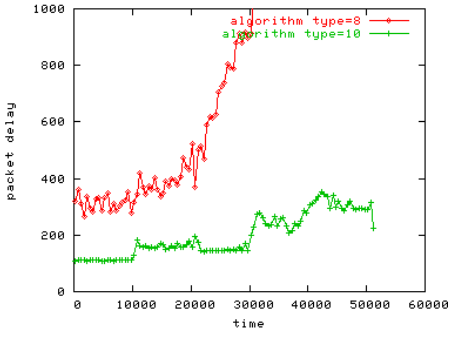


Fig. 6. Different speed vs. delay

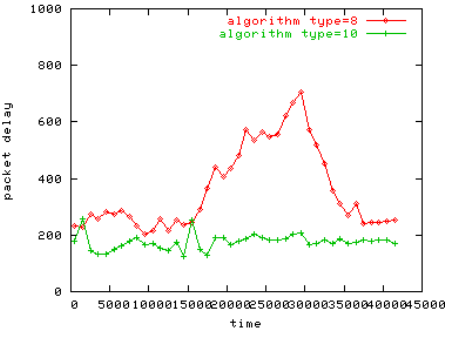


Fig. 7. Different probabilities vs. delay

reaches a small peak after changes of traffic characters (speed or probabilities), and turns to be smooth soon. But for Type-8, the delay changes dramatically according to the changes of traffic(see Fig. 8 and Fig. 9). It is obvious that the detection and adaptation of RPDS contribute to the performance.

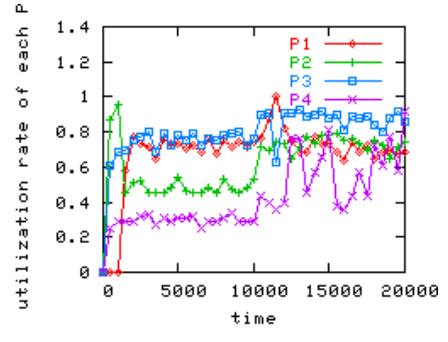


Fig. 8. Utilization rates of PEs

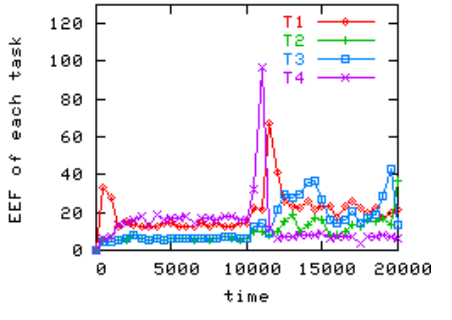


Fig. 9. *EEF* of tasks

5 Conclusions

The Replication-Based Partial Dynamic Scheduling (RPDS) is proposed in this paper. It tries to solve the problem of processing resources scheduling on the heterogeneous, fully-connected, and multi-threading NP hardware architecture. The main idea of RPDS algorithm is two-phase scheduling: static phase and dynamic phase. The static phase scheduling performs task pre-assignment before processing data-packets. It guarantees that each task could hold the minimal processing resources and keep the cost lowest. The dynamic phase scheduling occurs during the processing data-packets. When busy PEs and idle PEs coexist, the bottleneck task will be duplicated to the idle PE which makes the lowest cost. The future work includes the theoretic verification of RPDS, update of , and experiments to the default parameters, etc.

Acknowledgments. This work is partially supported by Graduate Starting Seed Fund of Northwestern Polytechnical University (No. M016634).

References

1. Malloy, B.A, Lloyd, E.L., Soffa, M.L.: Scheduling DAG's for Asynchronous Multiprocessor Execution. IEEE Trans. Parallel and Distributed Systems 5(5), 498–508 (1994)
2. Yan, S., Zhou, X., Wang, L., Wang, H.: GA-Based Automated Task Assignment on Network Processors. In: ICPADS 2005. Proc. of the 11th international Conference on Parallel and Distributed Systems, July 20–22, 2005, pp. 112–118. IEEE Computer Society Press, Los Alamitos (2005)
3. Franklin, M., Datar, S.: Pipeline task scheduling on network processors. In: Proc. of Third Network Processor Workshop in conjunction with Tenth International Symposium on High Performance Computer Architecture (HPCA-10), pp. 103–119 (February 2004)
4. Yang, L., Gohad, T., Ghosh, P., Sinha, D., Sen, A., Richa, A.: Resource mapping and scheduling for heterogeneous network processor systems. In: ANCS 2005. Proc. of the 2005 Symposium on Architecture for Networking and Communications Systems, pp. 19–28 (2005)
5. Ramaswamy, R., Weng, N., Wolf, T.: Application Analysis and Resource Mapping for Heterogeneous Network Processor Architectures. In: Proc. of Network Processor Workshop, Madrid, Spain, pp. 103–119 (2004)
6. Wolf, T., Pappu, P., Franklin, M.A.: Predictive scheduling of network processors. Comput. Networks 41(5), 601–621 (2003)
7. Weng, N., Wolf, T.: Pipelining vs. Multiprocessors-choosing the Right Network Processor System Topology. In: Proc. of ANCHOR 2004, Munich, Germany (2004)
8. Weng, N., Wolf, T.: Profiling and mapping of parallel workloads on network processors. In: Proc. of 20th ACM Symposium on Applied Computing (SAC) (March 2005)
9. Wolf, T., Weng, N., Tai, C.: Design considerations for network processor operating systems. In: ANCS 2005. Proc. of the 2005 Symposium on Architecture for Networking and Communications Systems, October 26–28, 2005, pp. 71–80. ACM Press, New York (2005)
10. Aggarwal, A., Franklin, M.: Instruction Replication for Reducing Delays Due to Inter-PE Communication Latency. IEEE Trans. Comput. 54(12), 1496–1507 (2005)

The Optimum Location of Delay Latches Between Dynamic Pipeline Stages

Mahmoud Lotfi Anhar[1] and Mohammad Ali Jabraeil Jamali[2]

[1] Islamic Azad University Khoy Branch , Khoy, Iran
MahmoudLotfie@gmail.com
[2] Islamic Azad University Shabestar Branch, Shabestar , Iran
m_jamali@itrc.ac.ir

Abstract. Latches are used between pipeline stages to get Minimum Average Latency (MAL). An optimization technique based on introducing a method to search the most proper location of noncompute delay latches between nonlinear pipeline stages is given. The idea is to find a new collision vector which is adaptable with pipeline topology and modifies reservation table, yielding MAL at minimum execution time. This approach not only reduces execution time of hardware, but also minimizes favorite collision vector search time.

Keywords: Latch, Pipeline, Minimum, Average, Latency, MAL, Reservation, Table, Collision.

1 Introduction

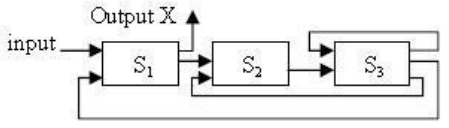
When scheduling events in a pipeline, the main objective is to obtain the shortest average latency between initiations without causing collisions. The pipeline utilization is limited by the collision characteristics which are the result of the usage patterns of the segments. One way of modifying usage pattern is by segment replication. Another way of changing a usage pattern is by inserting noncompute segments, which simply provide a fixed delay between some computation steps. A methodology has presented for modifying the collision characteristics with the insertion of delays so as to increase the utilization of segments by Davidson [1]. The purpose of delay insertion is to modify the reservation table, yielding a new collision vector. This leads to a modified state diagram, which may produce greedy cycles meeting the lower bound the MAL.

2 Single Function Pipelines

The combined set of permissible and forbidden latencies can be displayed by a j-bit binary vector C=($C_jC_{j-1}...C_1$). The value C_i=1 if latency i causes a collision and C_i=0 if latency i is permissible. Figure 1 shows an example of 3 stage pipeline and corresponding reservation table.

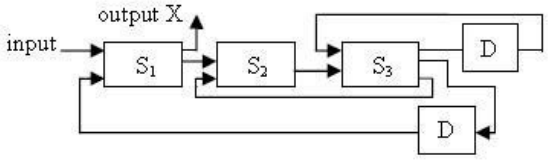
From reservation table MAL must be 2 but state diagram shows 3. Hence delay latches must be used. A solution proposed by Hwang [2]. Two latches are

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 14–17, 2007.



	Time				
	1	2	3	4	5
S_1	X				X
S_2		X		X	
S_3			X	X	

Fig. 1. A 3-stage pipeline and reservation table



	1	2	3	4	5	6	7
S_1	X						X
S_2		X		X			
S_3			X		X		
D				X			
D						X	

Fig. 2. Delay Insertion

added between S3-S3 and S3-S1 feedback loops and modified reservation table is presented in Fig. 2.

Our approach is shown in Fig. 3. It is 1 cycle faster than previous method.

3 Methodology

Assume a j-bit initial collision vector $X_j=(x_j\ x_{j-1}...x_1)$ which must be evaluated to produce {MAL} and $X_i=(x_i\ x_{i-1}...x_1)$ is initial collision vector before inserting latches . First we try evaluate X_j supposing it has a constant latency greedy cycle (G.C) equals the MAL.

Lemma 1. *If G.C=$\{k\}$ then $j \neq q.k$ where q is an integer . Since $x_j = 1$ and SHR[X_j , k] OR $X_j = X_j$, hence $x_k = x_{2k} = ... = 0$,and $j \neq \{k,2k,3k,...\}$.*

Time reduction considerations lead us to minimize j as it is possible. If $i+1 \neq q.k$ we start j with i+1 which is the best choice and yields minimum processing time else j= i+2 is the next choice.

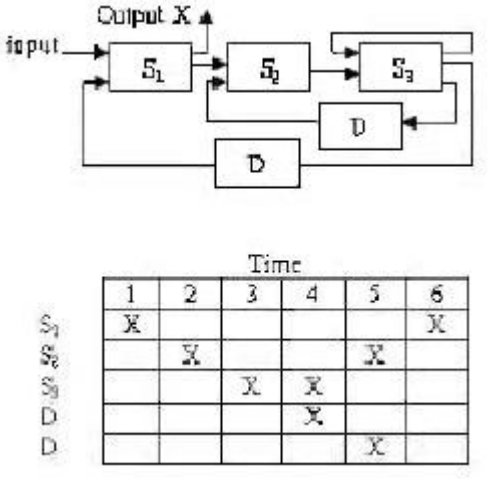


Fig. 3. Insertion delay latches using new approach

Example 1. In Fig. 1 X_i = (1011) and MAL is 2 then we can select j = 5. The new collision vector after inserting latches is X_j = (x_5 x_4 x_3 x_2x_1). Assume at this case MAL is 2, then

SHR[X_j , 2] OR X_j = X_j $\Longrightarrow$ $x_1 = x_3 = x_5 = 1$, $x_2 = x_4 = 0$.

Now examine X_j = (10101). Fig. 3 shows modified state diagram with a reduced MAL. It has better time response than previous approach.

Example 2. Fig. 3 is the reservation table which is used in [1]. Choosing X_j = $(x_7x_6x_5x_4x_3x_2x_1)$, G.C = {3} and using lemma 1 yields X_j= (1011011) and it is a possible solution.

Another way to find X_j is choosing a greedy cycle such as {k_1,k_2}.

Lemma 2. *If G.C = {k_1, k_2} then x_{k1}= x_{k2} = x_{k1+k2}= 0 and we can find two internal states (Y , Z)in state diagram satisfy following conditions:*

1. Y_{K1}=Z_{K2}=0.
2. Z= X OR SHR [Y,k_1].
3. Y=X OR SHR [Z,k_2].

Assume X= (x_j x_{j-1}... x_1) , Y=(Y_j Y_{j-1}...Y_1) and Z=(Z_j Z_{j-1}... Z_1) then
$Z_{j-(k1+i)}$ = $X_{j-(k1+i)}$ OR Y_{j-i+1} , i = 1 to j-(k_1+1).
$Y_{j-(k2+i)}$ = $X_{j-(k2+i)}$ OR Z_{j-i+1} , i = 1 to j-(k_2+1).
Z_{j-k1+i} = X_{j-k1+i} , i = 0 to k_1.
Y_{j-k2+i}= X_{j-k2+i} , i = 0 to k_2.
X_j=Y_j=Z_j= 1.
$Z_{j-(k1+1)}$= 1 , $Y_{j-(k2+1)}$=1
Then j-(k_1+1) $\neq$ k_2 and j-(k_2+1) $\neq$ k_1. j must satisfy following criteria :
j $\neq$ k_1+k_2+1 $\Rightarrow$ j$\geq$ k_1+k_2+2.

Example 3. Another solution to example1 is to choose greedy cycle {1,3}. Finding this greedy cycle is easy by introduced method. First select j = 1+3+2 =6 , then X = (1 x_5 0 0 x_2 0). Now use lemma 2 and find result : X = (1 0 0 0 1 0) , Y = (1 0 0 1 1 0) and Z=(1 1 0 0 1 1). Reservation table of new initial collision vector has 7 columns (1 cycle slower than previous X resulted with {2} greedy cycle).

In this case MAL = $(k_1 + k_2)/2$ then X must be calculated for following set :1, k_1+k_2-1, 2, k_1+k_2-2 , ,$(k_1+k_2)/2$ -1,$(k_1+k_2)/2$ + 1. All of these choices have MAL = $(k_1+k_2)/2$ and number of them is $(k_1+k_2)/2$ - 1. Then we calculate X for constant greedy cycle MAL first, and if it is not sufficient, propose one of above introduced non-constant MALs.

4 Algorithm

The following algorithm is used in this method:

1. For each MAL generate a j-bit collision vector X_j which $j = i + 1$ (i is the number of initial collision vector X_i bits). Assume greedy cycle is MAL.
2. Test X . If X satisfies then quit, else $j = j + 1$;
3. Try 2 until j < k_1+k_2+2 .
4. Select one of non-constant previously stated greedy cycles. Select another greedy cycle if does not satisfy criteria stated in lemma 2.

5 Concluding Remarks

We have presented a new method to evaluate initial collision vector using constant and non-constant greedy cycles. For increasing throughput and then reducing processing time we began with constant greedy cycles. Then other non-constant cycles related to MAL is used. This method not only reduces processing time but also is a faster one than other methods.

References

1. Patel, J.H., Davidson, E.S.: Improving the Throughput of a Pipeline by Insertion of Delays. Coordinated Science Lab, University of Illinois, Urbana, Illinois 61801, 132–137 (1976)
2. Hwang, K.: Advanced Computer Architecture: Parallelism, Scalability, Programmability. McGraw-Hill, New York (1993)
3. Davidson, E.S., Thomas, D.P., Shar, L.E., Patel, J.H.: Effective Control for Pipelined Computers. In: Proc. COMPCON, pp. 181–184 (1975)
4. Kogge, P.M.: The Architecture of Pipelined computers. Mc-Graw-Hill, New York (1981)

A Novel Fault-Tolerant Parallel Algorithm

Panfeng Wang, Yunfei Du, Hongyi Fu, Haifang Zhou,
Xuejun Yang, and Wenjing Yang

National Laboratory for Paralleling and Distributed Processing,
College of Computer, National University of Defense Technology,
Changsha, Hunan, 410073, China
{wpfeng,forest80,mrfool_163,haifang_zhou,xjyang}@nudt.edu.cn

Abstract. The mean-time-between-failure of current high-performance computer systems is much shorter than the running times of many computational applications, whereas those applications are the main workload for those systems. Currently, checkpoint/restart is the most commonly used scheme for such applications to tolerate hardware failures. But this scheme has its performance limitation when the number of processors becomes much larger. In this paper, we propose a novel fault-tolerant parallel algorithm FPAPR. First, we introduce the basic idea of FPAPR. Second, we specify the details of how to implement a FPAPR program by using two NPB kernels as examples. Third, we theoretically analyze the overhead of FPAPR, and find out that the overhead of FPAPR decreases with the increase of the number of processors. At last, the experimental results on a 512-CPU cluster show the overhead introduced by the algorithm is very small.

Keywords: high-performance computing, fault tolerance, parallel algorithm.

1 Introduction

There is a trend that the high-performance computing systems are consisted of more and more processors. The fastest one, IBM Blue Gene/L, has 131,072 processors, and even the smallest computer system in the Top10 has 9024 processors. However, as the complexity of a computer system increases, its reliability is drastically deteriorating. For example, if the reliability of individual components is 99.999%, then for the whole system consisting of 100,000 non-redundant components, the reliability is nothing more than $(99.999\%)^{100000} = 36.79\%$. Such a low reliability is unacceptable in most applications.

A critical issue for machines of large size is the mean time between failures. Projecting from the existing supercomputers, a 100,000 processors supercomputer could see a failure every few minutes. But the applications running on these supercomputers are typically compute-intensive and will take a long time. Therefore, it is an important ability for computer systems consisting of such a large amount of processors to deal with processor failures.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 18–29, 2007.

Today, applications typically deal with process failures by writing out checkpoints periodically[7,3]. All processors save their computation state to a storage server periodically during the execution. Even if there is no failure, the overhead is inevitable. If a fault occurs, then all processes are forced to stop and the job is reloaded from the last checkpoint. For a large-scale system, checkpoint/restart may not obtain an effective utilization of the resources.

In this paper, we propose a novel **F**ault-tolerant **P**arallel **A**lgorithm based on **P**arallel **R**ecomputing (FPAPR for short). If there is no failure, the overhead of FPAPR is insignificant. If there is a failure, the overhead of FPAPR decreases with the increase of the processor number.

The rest of the paper is organized as follows. Section 2 is an introduction of related works. Section 3 introduces our scheme, FPAPR. In Section 4, we give a detailed presentation on how to write a FPAPR application by using two examples, EP and DT from NAS Parallel Benchmark. In Section 5, we evaluate the performance overhead of our fault tolerance algorithm in theoretic. Section 6 is the experimental results. Section 7 we conclude the paper and discuss future work.

2 Related Work

Checkpoint/restart scheme is the most typical fault tolerance handling technique for large-scale parallel computing. The conventional checkpoint/restart scheme uses the coordinated checkpoint protocol that the system saves a global checkpoint to a central file server at very coarse intervals (for example, once every hour or day). Such a checkpoint/restart scheme is straightforward and has been discussed and implemented elsewhere [9,10,8,4,5,2]. But this checkpoint/restart scheme has its performance and conceptual limitations. Firstly, the computation state of each processor has to be stored periodically to the disk of a special checkpoint server during execution. The I/O bandwidth of the checkpoint server will heavily impact the performance of whole system, whereas the I/O bandwidth is often hardly improved. When the amount of all checkpoint files is huge, the overhead of checkpoint will become unacceptable. Sometimes, the time to do a checkpoint or a restart will be longer than the time between failures. As a result, the conventional checkpoint/restart scheme becomes completely impractical as a means to deal with failure. Secondly, checkpoint/restart scheme has considerable overhead even if no failure occurs, which will significantly reduce the speedup of a parallel application. Thirdly, compared with the processor and the memory, the hard disk is least dependable. The dependability of whole system will decrease when extra hard disks are involved to save checkpoint files.

Whatever point of view we are from, the dependability or the speed of development, the processor is the best, the memory is the secondary, and the hard disk is the worst. In contrast to the above works that need extra hard disks or memories, FPAPR tolerates a processor failure by using one or more remaining processors to recompute the subtask of the failed one. The main advantage of FPAPR lies in two points. Firstly, it is a scalable algorithm. Secondly, it

has almost no overhead in the failure-free condition. The details of FPAPR are described in the next section.

3 FPAPR

There is a common requirement for fault tolerance that the data of the failed process must be accessible by other processes. For example, checkpointing techniques usually require the checkpoint file of the failed process being saved in a common server, so the restarted process can read it. This requirement is easy satisfied for SPMD parallel programs which are prevalent in high performance computing. In many SPMD parallel programs, the input data of each process is determined by its process rank. As long as surviving processes know who fails, they can compute or reload the input data of failed one and complete the workload of it. In this paper, our FPAPR is mainly designed for this kind of parallel application.

3.1 The Basic Idea of FPAPR

In fact, many parallel applications have inherent ability for fault tolerance. For a SPMD application, $1/N$ of the whole workload is distributed to each processor. If any one is failed, we can use multiple remaining processors to share the subtask of the failed one. This is the basic idea of FPAPR. We show the task partitioning of FPAPR in Fig.1.

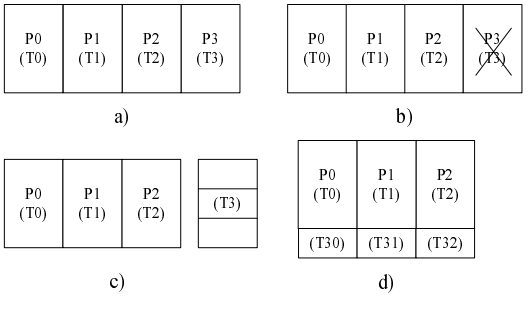


Fig. 1. Task partitioning of FPAPR

In Fig.1 a), the whole task is partitioned to four processors. For process P_i, it will take time T_i to perform its subtask. If the processor P_3 fails, as shown in Fig.1 b), then its subtask is repartitioned to the remaining three processors in Fig.1 c). As a result, the execute time of each remaining processor P_i becomes $T_i + T_{3i}$ in Fig.1 d).

We illustrate the execute process of one processor in a FPAPR program in Fig.2, where we assume that the program includes a computation segment and

a communication segment, and the whole computation consists of a serial part W_S and a parallel part W_P. In the failure-free condition, the subtask of each processor will be $W_S + \frac{W_P}{N}$, and there is no extra overhead. If there is one failure, the subtask of each remaining processor will be $W_S + \frac{W_P}{N} + W_S + \frac{W_P}{N(N-1)}$, and the extra overhead of FPAPR is $W_S + \frac{W_P}{N(N-1)}$.

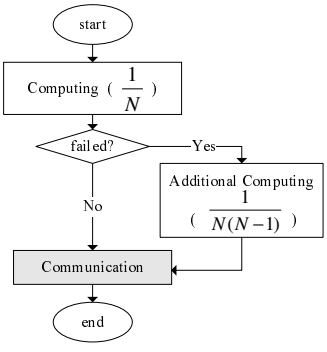


Fig. 2. The execution process of FPAPR

For a parallel application, each subtask is unique. If any processor fails which takes a subtask on, the whole application will fail too. In order to tolerance a processor failure, we have to firstly recovery the lost data in the failed one. Checkpoint/restart scheme perform the work by saving the data periodically to a storage server, but FPAPR find the lost data back through another way, i.e. recomputing it in parallel by multiple surviving processors. If the lost data can be computed in parallel, the overhead of recomputation will decrease with the increases of N.

3.2 Fault Processing in FPAPR

Fault processing typically includes three steps: detection, information and recovery. Here, we assume that parallel computing platform, such as MPI, provides failure detection and information (One such implementation of MPI is FT-MPI[6]), and at most one failure occurs during execution. FPAPR focuses on how to recover computing using multiple surviving processors to recomputing the subtask in the failed processor in parallel. There are two main issues: when to recompute and how many processor the subtask is repartitioned to.

There are two strategies for the first issue. One is a studious strategy. Once a process fails, the parallel computing platform informs the application immediately of it and the application turns to the fault processing module on the instant. The other is a lazy strategy. Only when the lost data is needed by the other processors, it is recomputed. The former is straightforward but is difficult to implement. Using the latter strategy, the application just needs to check failure in a few key points where the fault will be carried to another processor. The

fault must be processed before the statement, which will carry data from or to the failed processor, is executed. In a MPI application, the fault spreads through MPI calls such as MPI_Send or MPI_Bcast. So the application should check and process fault before these statements.

For the second issue, we think that in a scalable parallel application, the subtask of failed processor is also computation-intensive, and it can be repartitioned to all remaining processors to shorten the recovery time. Fig.2 shows that all $N-1$ remaining processors are used to recompute the subtask on the failed one. However, in some special parallel application, the subtask of failed processor may be very simple, and using $N-1$ processors does not benefit much in comparison with using one. So without any performance loss, dispatching the subtask of failed processor to one of the remaining processors is simpler. In the next section, we will illustrate how to use implement FPAPR in these two kinds parallel programs.

4 Examples of FPAPR Implementation

FPAPR is an application-based fault-tolerant scheme. Fault processing in FPAPR is closely related to specific applications. We have implemented fault-tolerant versions of MPI-based EP and DT kernels from NAS Parallel Benchmark (NPB3.2.1-MPI). In this section, we provide an overview of how each kernel works and how we make it fault-tolerant.

We choose MPICH2-1.0.5 as the parallel computing platform and simulate a fail-stop failure by killing a process. And we added two global variables *HasFailed* and *FailedRank* to MPICH2-1.0.5, thus enabling processes to obtain the rank of the failed process through the failure detection interface.

4.1 EP

EP (Embarrassingly Parallel) generates pairs of Gaussian random deviates according to a specific scheme and tabulates the number of pairs in successive square annuli [1]. EP is a typical data-parallel SPMD application. Each processor executes one part of the whole task independently, and communications occur only after computing.

In Fig.3 a), we show the execution process of NPB EP in one processor, and in Fig.3 b) we illustrate the execution process of the FPAPR version.

There are four main steps in NPB EP: 1) Task partitioning, 2) Main computing loop, 3) Collecting global results through communication, 4) Results reporting.

Each processor runs independently in step 1 and 2.The failure of one processor does not infect the others, so fault processing is not necessary. The global results are collected through collective communications (MPI_Allreduce) in step 3. Faults will spread with messages. So this is a fault processing point. It is necessary to check whether each collective communication returns successfully. If not, the program should turn to the fault processing module.

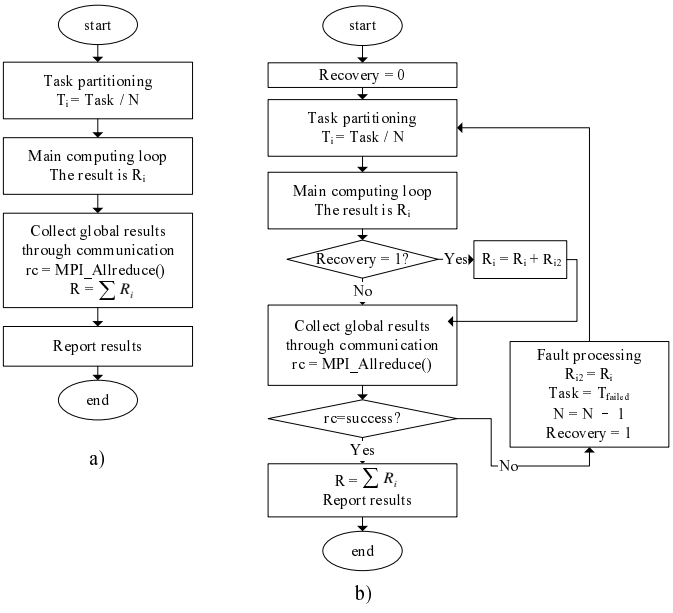


Fig. 3. The execution process of EP

In the fault processing module, each remaining processor must save the current local result ($R_{i2} = R_i$), set the program enter into recovery state ($Recovery = 1$), and change the parameters of task repartitioning module ($Task = T_{failed}, N = N - 1$). Then the execution reenters into step 1 and 2, and gets a new result R_i. Before step 3, the two results are cumulated ($R_i = R_i + R_{i2}$). Thus, the execution goes back to the normal way, collects global results through collective communications, and reports final results. The processor of rank 0 is the default one for reporting final results in NPB EP. In FPAPR version, we use the processor of rank 1 to report the final results if rank 0 fails.

4.2 DT

Different from EP, DT (Data Traffic) is typical communicate-intensive benchmark. It includes data generating nodes (sources), data processing nodes (comparators), data consuming nodes (sinks), communication graph, and two special nodes "Launch" and "Report". The arcs of the graph indicate the directions the data communicated between the nodes. Each source uses its own seed and generator to generate a random number of feature points, which size scale from a few KB(class S benchmark) to many MB(class C benchmark). The data stream from/to the processing nodes according to the communication graph. The computation in each node is very simple and can be rapidly completed. Shuffle network is the most complicated communication graph. In Fig. 4 a) we illustrate

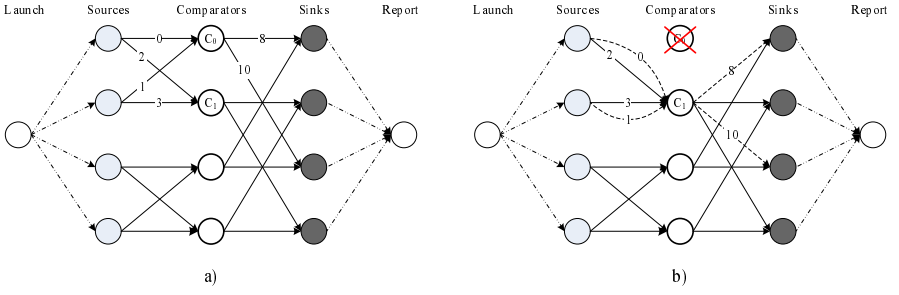


Fig. 4. Data communication graph of DT

the communication graph of class S, and in Fig. 4 b) show the FPAPR version with one failure.

As shown in Fig.4, comparator C_0 receives messages 0 and 1, processes them, and sends results to two sinks through messages 8 and 10. When C_0 fails, the others should partition the subtask on it. Since this subtask is very simple, it is directly dispatched to C_1 rather than all the $N-1$ remaining processors.

We implement a function, *who*(C_i), to choose a substitute for C_i, and replace the destination/source parameter in MPI communication called by C_i. If C_i fails, data will be sent to or receive from the substitute of C_i chosen by function *who()*. Figure 5 gives an example of using this function.

```
int SendResults(DGraph *dg,DGNode *nd,Arr *feat){
  ...
  if (head->address != nd->address){
    // MPI_Send(&feat->len,1,MPI_INT,head->address,tag,MPI_COMM_WORLD);
    // MPI_Send(feat->val,feat->len,MPI_DOUBLE,head->address,tag,MPI_COMM_WORLD);
    MPI_Send(&feat->len,1,MPI_INT,who(head->address),tag,MPI_COMM_WORLD);
    MPI_Send(feat->val,feat->len,MPI_DOUBLE,who(head->address),tag,MPI_COMM_WORLD);
  }
  ...
}
```

Fig. 5. An example of using the function who()

5 Performance Analysis of FPAPR

In this section, we analyze the scalability and extra overhead of FPAPR. It is assumed that the whole workload, W, is measured by time and includes a serial part W_S and a parallel part W_P, i.e. $W = W_S + W_P$. Let $f = W_S/(W_S+W_P)$. We suppose that the workload can be scaled to $W_S + G(N)W_P$ in a parallel system consisting of N processors. The overhead of parallel computing is W_O. There are some extra overhead in a FPAPR program. For example, when a processor

failed, the MPI communicator (such as MPI_COMM_WORLD) is corrupted. It is necessary to recovery it in the fault processing module. Let W_{ft} represent such a overhead of FPAPR.

The serial part can not be speed up by $N-1$ processors. Thus, if one processor fails, the total overhead of FPAPR is

$$O_{FPAPR} = W_S + \frac{G(N)W_P}{N(N-1)} + W_P. \tag{1}$$

The percentage overhead of FPAPR is:

$$PO_{FPAPR} = \frac{W_S + \frac{G(N)W_P}{N(N-1)} + W_{ft}}{W_S + \frac{G(N)W_P}{N} + W_O} \tag{2}$$

If the overheads (W_O and W_{ft}) and f are so small that can be ignored, then we have:

$$PO_{FPAPR} = \frac{1}{N-1} \tag{3}$$

From Eq.(1), we conclude that the overhead of FPAPR decreases with the increase of N. From Eq.(3), we know that in some special conditions, the percentage overhead equals $1/(N-1)$. These two equations imply the same conclusion that FPAPR is scalable. Then, we discuss the speedup of FPAPR with one failure.

The time of FPAPR program executing in a single processor is $T(1)$. If a processor failure occurs, $T(1)$ is infinite. For a FPAPR program, at last one remaining processor is needed to perform the whole task. So $T(2)$ is the baseline for comparison. We make $T(1)$ equal $T(2)$. Then $T(1)$ is

$$T(1) = W_S + \frac{G(N)W_P}{2} + W_S + \frac{G(N)W_P}{2(2-1)} + W_{ft} = 2W_S + G(N)W_P + W_{ft} \tag{4}$$

And $T(N)$ is

$$\begin{aligned} T(N) &= W_S + \tfrac{G(N)W_P}{N} + W_O + W_S + \tfrac{G(N)W_P}{N(N-1)} + W_{ft} \\ &= 2W_S + \tfrac{G(N)W_P}{N-1} + W_O + W_{ft} \end{aligned} \tag{5}$$

Using two equations, we obtain the following speedup equation:

$$S = \frac{T(1)}{T(N)} = \frac{2W_S + G(N)W_P + W_{ft}}{2W_S + \frac{G(N)W_P}{N-1} + W_O + W_{ft}} \tag{6}$$

Notice that $W = W_S + W_P$ and $f = \frac{W_S}{W_S+W_P}$, and then Eq.(6) can be changed into:

$$S = \frac{T(1)}{T(N)} = \frac{2f + G(N)(1-f) + \frac{W_{ft}}{W}}{2f + \frac{G(N)(1-f)}{N-1} + \frac{W_O+W_{ft}}{W}} \tag{7}$$

Equation(7) is similar to Sun & Ni equation[11].

If $G(N)$ equals N, and W_O, W_{ft} can be ignored, then an ideal speedup can be attained:

$$S = \frac{T(1)}{T(N)} = \frac{2f + N(1-f)}{2f + \frac{N(1-f)}{N-1}} = \frac{2f + N(1-f)}{1 + f + \frac{1-f}{N-1}} \approx \frac{2f}{1+f} + N(1 - \frac{2f}{1+f}) \quad (8)$$

From Eq.(8), we conclude that when N is large enough and all overheads (W_O and W_ft) can be neglectable, the ideal speedup of FPAPR is proportional to N. If $G(N)$ is not equal to N, but the serial part W_S is very small in comparison with the whole workload, i.e. , W_O, W_ft can be ignored too, then we can obtain another helpful equation:

$$S = \frac{T(1)}{T(N)} = \frac{G(N)}{\frac{G(N))}{N-1}} = N - 1 \quad (9)$$

Equation(9) means that for some applications which are very suitable for data-parallel programming model, the speedup of FPAPR just subtracts one when one processor fails. If there is no processor failure, the overhead is maybe different between FPAPR EP and FPAPR DT. There are some collective communication calls in EP. If one process in a MPI communicator (such as MPI_COMM_WORLD) fails, all the collective communication calls in this communicator will fail too. So some extra recovery operations about the MPI communicator are needed to protect these collective communication calls. The overhead of these operations maybe relates to the number of processors. The communication calls in DT are much simple, and only peer-to-peer communication calls such as MPI_Send and MPI_Recv are involved. So the overhead of FPAPR DT mainly lies in the extra messages processing for the substitute of the failed processor.

6 Experiment Results

We implement EP and DT described in section 4 on a 256-node cluster. Each node has two XEON 3.2GHz CPUs and 4GB RAM.

We run three sets of tests for each program. The first one is a pure NPB version with no failures, named NPB EP. The second is a FPAPR version, but there is no failure, named FPAPR-EP(0). And the third is a FPAPR version, injected randomly with one processor failure, named FPAPR-EP(1).

6.1 EP

The implementation results are illustrated in Figure 6∼8. Fig.6 and Fig.7 show the execution time and overhead percentage respectively. Figure 8 gives the speedup of FPAPR-EP.

From Fig.6 and Fig.7, we can see that the overhead of FPAPR EP with 1 failure decreases as the number of processors increases. The right reason is that NPB EP is a typical compute-intensive application, and the proportion of the serial part and extra overheads are very small. Therefore, the prior conditions

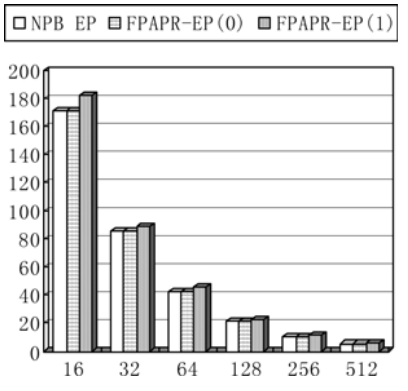


Fig. 6. Execution Time

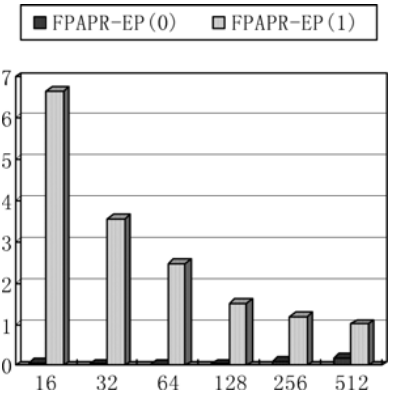


Fig. 7. Overhead percentage

expressed by Eq.(3) and Eq.(9) are nearly satisfied. And we can see the results shown in Fig.7 and Fig.8 accord with those indicated by Eq.(3) and Eq.(9). The overhead percentages approximate to $1/(N-1)$. And the speedups of FPAPR EP with one failure approximate to $N-1$.

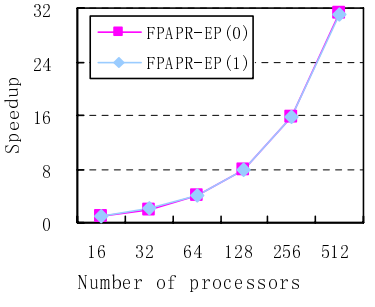


Fig. 8. Speedup

Fig.6 and Fig.7 also show that the overhead of FPAPR EP without failures keeps very slow. These are in line with our expectation. But the overhead percentages grow with the increase of N. This can be explained that the execution time of those recovery operations of the MPI communicator such as MPI_Comm_dup grows when the number of processes in the communicator increases.

6.2 DT

The results of DT are listed in Table 1 and Table 2.

DT is a special instance for FPAPR. The basic idea of FPAPR is utilizing the computing power of multi-processors to speed up the recovery of the lost data of

Table 1. Execution Time (s)

CLASS	NPB DT	FPAPR DT	
		No failure	1 failure
S	0.0600	0.0632	0.0648
W	0.6420	0.6681	0.6860
A	9.1892	9.2500	9.3875

Table 2. Overhead Percentage of FPAPR Version (%)

CLASS	No failure	One failure
S	1.0533	1.0800
W	1.0407	1.0685
A	1.0066	1.0216

the failed process. But the main workload of DT is the communication part, and there is no chance of using the computing power of multi-processors. However, the experimental results show that the overhead percentages in both circumstances are very small. This implies that FPAPR can be applicable in broader areas.

7 Conclusions and Future Work

We have given a novel fault-tolerant algorithm based on parallel recomputing, FPAPR, for parallel computing. This algorithm enables an application designed as SPMD programming model to tolerate a single processor failure without introducing any hardware overhead. As long as the number of processors N is larger than 2, the computation will not be terminated by a single processor failure. For a data-parallel application, the overhead of fault-tolerance decreases and the speedup grows with the increase of number of processors.

We have implemented this algorithm in two NPB kernels and give performance results on a 512-CPU cluster. The results are gratifying. The overheads of FPAPR without failure are so small that they can be neglected. For a computation intensive SPMD program like EP, the overhead of FPAPR with a single failure is nearly in reverse proportion to N-1. For a communication intensive SPMD program like DT, the overhead of FPAPR with a single failure is also very small.

Our continuing progress with this work has been in two directions. First, we are researching the applicability of FPAPR in broader areas and finding out the common character of applications whose FPAPR versions are very efficient. Second, we will implement an automatic source-to-source tool based on compiler technique for translating a common parallel program into its FPAPR version.

Acknowledgments. This work was supported by NSFC (60621003 and 60603081).

References

1. Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V., Weeratunga, S.: The nas parallel benchmarks. Technical report (1994)
2. Bronevetsky, G., Marques, D., Pingali, K., Stodghill, P.: Automated application-level checkpointing of mpi programs. In: PPoPP 2003. Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming, San Diego, California, USA, pp. 84–94. ACM Press, New York, NY, USA (2003)
3. Chiueh, T.-C., Deng, P.: Evaluation of checkpoint mechanisms for massively parallel machines. In: FTCS 1996. Proceedings of the The Twenty-Sixth Annual International Symposium on Fault-Tolerant Computing, Washington, DC, USA, p. 370. IEEE Computer Society Press, Los Alamitos (1996)
4. Mootaz Elnozahy, E.N., Alvisi, L., Wang, Y.-M., Johnson, D.B.: A survey of rollback-recovery protocols in message-passing systems. ACM Comput. Surv. 34(3), 375–408 (2002)
5. Engelmann, C., Geist, A.: Super-scalable algorithms for computing on 100,000 processors. pp. 313–321 (2005)
6. Fagg, G.E., Dongarra, J.: Ft-mpi: Fault tolerant mpi, supporting dynamic applications in a dynamic world. In: PVM/MPI, pp. 346–353 (2000)
7. Geist, A., Engelmann, C.: Development of naturally fault tolerant algorithms for computing on 100,000 processors (2002)
8. Plank, J.S.: Improving the performance of coordinated checkpointers on networks of workstations using RAID techniques. In: 15th Symposium on Reliable Distributed Systems, pp. 76–85 (October 1996)
9. Plank, J.S., Li, K.: ickp: A consistent checkpointer for multicomputers. IEEE Parallel Distrib. Technol. 2(2), 62–67 (1994)
10. Stellner, G.: CoCheck: Checkpointing and Process Migration for MPI. In: IPPS 1996. Proceedings of the 10th International Parallel Processing Symposium, Honolulu, Hawaii (1996)
11. Sun, X.-H., Ni, L.M.: Another view on parallel speedup. In: Supercomputing 1990. Proceedings of the 1990 conference on Supercomputing, New York, New York, United States, pp. 324–333. IEEE Computer Society Press, Los Alamitos, CA, USA (1990)

The Design on SEU-Tolerant Information Processing System of the On-Board-Computer

Huang Ying[1,2], Zhang Chun-yuan[1], Liu Dong[1], Li Yi[1], and Weng Sheng-xin[2]

[1] Department of Computer Science, National University of Defense and Technology, Changsha, 410073 Hunan Province, P.R. China
[2] Department of Computer Management Center, Navy General Hospital, Beijing 100037, P.R. China
{cyzhang,yinghuangying}@nudt.edu.cn

Abstract. For SEU(Single-Event-Upsets) of space radiation environment, a multi-level fault-tolerant mechanism based-on FPGA, which has greatly improved the system's ability of resisting SEU was presented. The design of three-level fault-tolerant included the dual fault-tolerant system based-on FPGA, the module-level triple modular redundancy, and the chip-level SEU-tolerant FPGA. Finally, the evaluation for the SEU reliability of the OBC(on-board-computer) was mentioned.

Keywords: Single-Event-Upsets, Field Programmable Gate Array, Dual Fault-Tolerant, Triple Module Redundancy, Cost-Off-The-Shelf.

1 Introduction

Nowadays, the R&D(research and development) and application of the COTS-based (Cost-Off-The-Shelf) space computer system are the keystone of many universities and institutes all around the world. Moreover, as a combined product of spatial flight application and computer, one of the targets of designing COTS-based computer subsystem is to achieve low-cost, light-weight and short-development-cycle, and it is supported by using commercial device and development kits[1].

Due to the high energy particles and diversified external interferences in the spatial environment, the safty of flight, at a large extent, relies on the design of computer system reliability. In order to avoid spatial fault, the selection of devices should be seriously considered. Normally, we adopt industrial-class and commercial-class chips in the spatial COTS-based computer. However, from the view of system-design level, the most important thing is to introduce the fault-tolerant technique. Combined with the system architecture and needs of the design, we put forward a design scheme of COTS-based multi-level fault-tolerant architecture, and finally used an analytical model to validate the reliability of the design.

2 Architecture

The COTS-based space computer system architecture is shown in figure 1. The COTS-based chips of the system like SRAM, Flash, CPU and so on, are all prone to suffer from SEU, and consequently result in bad working status.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 30–39, 2007.

SEU is caused by electrification particles casting into the sensitive fields of integrate circuit[2]. It usually introduces many important errors such as CPU's inner register content changed or bit flipped. These errors may directly result in wrong computing results, wrong sequence of program executing, and even breakdown of the system. In order to adapt to spatial radiation environment and improve system's reliability of SEU-mitigation, according to the rules of design on COTS-based devices, we put forward a multi-level SEU-mitigation design scheme based on FPGA(Field Programmable Gate Array). There were FPGA-based dual fault-tolerant system at the system level, FPGA-based SEU-mitigation design of RAM and Flash at the module level and SEU-mitigation design on SRAM-based FPGA at the chip level. As a result, the system's reliability to mitigate SEU has been improved by this multi-level-tolerant mechanism.

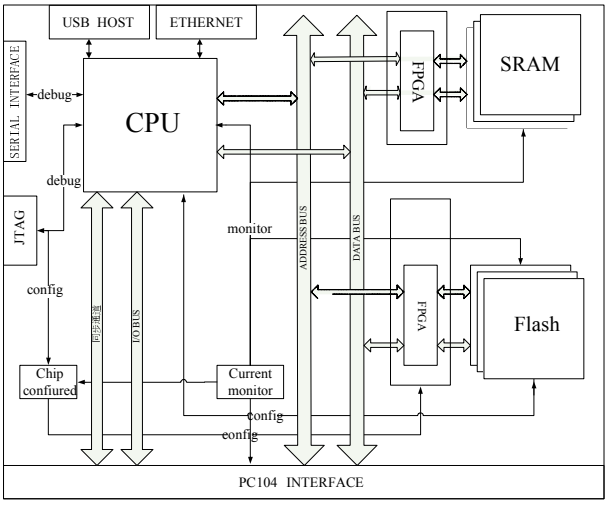


Fig. 1. The two same single boards were connected by the interface PC104 constituted of the dual fault-tolerant subsystem of the OBC

3 Design of Multi-level Fault-Tolerant System Architecture

3.1 System-Level Design of SEU Mitigation

This section introduces a kind of FPGA-based warm-back-up dual fault-tolerant system scheme. By efficiently integrated advantages and characteristics of FPGA chips, we not only implemented the mechanism of judging the fault computer of dual fault-tolerant system and resume by itself, but also arbitrated diversified data signals by MS(master symbol), so that the system would be immune to the influence of single-computer faults and hold communicating with external data.

The architecture of the adopted system-level dual fault-tolerant mechanism based on FPGA, is shown in Fig.2. The watchdog timer circuit(WDT0,WDT1) and the arbitrator(ARBITER) , which were both the key modules of the dual system,

respectively took charge of supervising its running states and arbitrating signals of the dual system like communication and control. In order to implement the system-level dual warm-back-up fault-tolerant function, the two CPU were linked by the middle interface module(the arbitrator). At the same time, FPGA was also responsible for providing the interface (arbitrated by FPGA) communicated with the external. It was supposed that the mechanism of CPU1 and WDT1 were respectively the same as CPU0 and WDT0. There are symbol explained as follows.

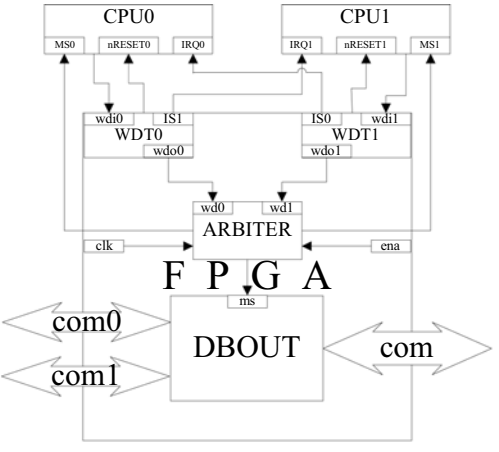


Fig. 2. The architecture of dual fault-tolerant system based on FPGA

A. CPU0 saw after 4 kinds of signals

①CPU0 periodically sent impulse signal wdi0 to WDT0, denoting that CPU0 works normally. ②CPU0 received nRESET from WDT0, in order to reset CPU0 and resume from fault state. ③CPU0 received IS0 from WDT1 to take over the system when CPU1 had been broken. ④CPU0 received MS0 simultaneously from arbitrator, in order to judge the respective current state of CPU0 and CPU1.

B. WDT0 took charge into inspecting CPU0

After the system had been electrified, CPU0 periodically sent WDT0 an impulse signal wdi0, which reset the counter of WDT0 to 0. When the next clk0(WDT0's inner clock) came, the counter of WDT0 recounted from 0. If WDT0 counted over some value(CPU0 had not sent wdi0 during a scheduled period), CPU0 was considered broken, and WDT0 would act as follows: ①WDT0 sent IS1 to CPU1, required it taking over CPU0 and kept system go on. ②WDT0 sent CPU0 nRESET0 to reset it. ③WDT0 sent low current-flow of wdo0 to the port wd0 of the ARBITER module of FPGA, in order to notify the arbitrator that CPU0 had been broken and should power on CPU1 to take the place of CPU0.

C. ARBITER was the core of the realization of dual warm-back-up mechanism

It judged the value of ms(master symbol) through changes of wd0 and wd1, and export the value of ms to DBOUT. When the system was powered on, the enable

signal ena set ms=0, wd0=0, and wd1=0, export ms=0, denoting that CPU0 was the host. After a period of time, if CPU1 had been broken(wd1=1), ms maintained value 0. If CPU0 had been broken, wd0 from WDT0 was set to 1, and simultaneously ms was set to 1. However, after CPU0 resumed from fault state, wd0 would be set to 0, and the value of ms would not be changed immediately. By the time CPU1 had been broken(it means that the value of wd1 had been changed to 1), the value of ms would be changed. In this way, it could be avoid to frequently switch the host and the guest from the dual system, by too frequently changed the value of ms. CPU could judge the host or guest from the dual system through the value of correspond master symbol MS.

D. The effect of DBOUT module

By judging the value of ms, the system port com was switched to communicate with correspond channels of each CPU. When system was powered on and initialized, com was switched into com0, and then CPU0, as the host, was responsible for communicate with external data out of system. If CPU0 appeared fault(namely ms=1), com was switched into com1, and CPU1 took instead of CPU0. The way to switch system communication channels by judging the value of ms, assured system was able to resume itself from fault state and continued to communicate with external data despite that a computer was broken in the dual system.

3.2 The Module-Level Mitigation Design on SEU

As memory device, SRAM and Flash are both prone to suffer from SEU. Usually there are two methods to mitigate SEU on memory in the module-level, EDAC(Erro Detection And Correction) and TMR(Triple Module Redundancy). With different coding measures, the capacity of fault tolerance are different. EDAC is a software fault-tolerant technique. But usually its ability to correct errors is confined to the cost and performance. And it costs much more additional delay time to check and correct errors, since comparatively opacity to the processor. Especially for high-speed device like SRAM, performance is influenced, and the complexity of design and difficulty of realization are increased as well. In this paper, we have adopted and implemented FPGA-based TMR technique to mitigate SEU on SRAM and Flash, due to the rich storage resource and based on the hypothesis that the possibility of SEU is 0 in the same bit of two memory cell.

3.2.1 SRAM-Mitigation Design

To improve the reliability at the maximum extent, and to reduce memory cells' upsets caused by high energy particles, we used three identified memory cells arbitrated by a controller to form a TMR fault-tolerant mechanism. The cell could be arbitrated out by TMR arbitrator, when it met SEU, and the system continued working freely. For the adequate memory resources, the upset cells could be scrubbed into right value instead of wrong[3]. Fig.3 shows the logic circuit of TMR.

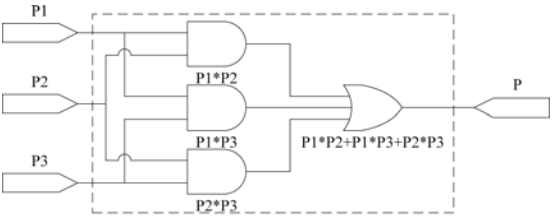


Fig. 3. The logic circuit of TMR

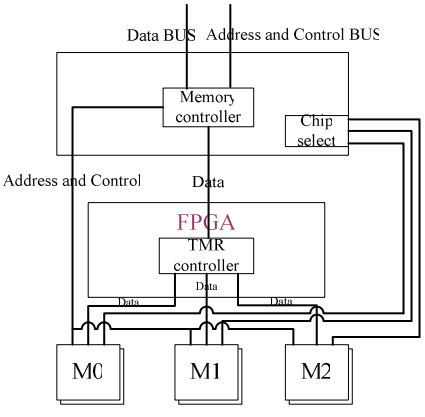


Fig. 4. The function of the arbitrating controller of TMR

Fig. 4. illustrates the function of the FPGA-based TMR arbitrator controller.

As Shown in the Fig.4, triple voting is typically used for functional blocks with high clock rates of at least 66 MHz between a processor and banks of volatile memory. To maximize the processor performance, this mechanism must allow for zero wait-state operation such that the fetch and write performance is not degraded for additional delays in implementing this voting methodology. From the processor perspective, triple voted memory is therefore achieved to perform the voting function within the same CPU read-memory request transaction; it is completely transparent to the processor. The general procedure to operate a triple voting memory mechanism is shown in Fig.5 [4] .

A typical triple voted controller was implemented in a radiation-tolerant FPGA and included the memory data integrity logic and circuitry, which were needed to enhance system reliability in space environment applications.

3.2.2 The SEU-Mitigation Design on Flash

Flash was used to reserve programs and data of the system while working. In order to assure correctness of the data, FPGA-based TMR technique was used to mitigate SEU. Data can directly be read out and written back by Flash(without memory controller), so it could be easily immune to SEU as long as TMR arbitrating mechanism. One of the advantages was transparent for the processor to directly access and execute instructions from Flash, without the processor judging whether it was correct.

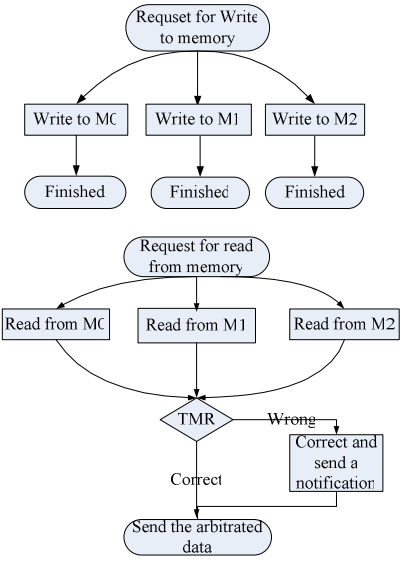


Fig. 5. The mechanism of memory controller of TMR

3.3 The Chip-Level Mitigation Design on SEU

The arbitrator of dual system and TMR arbitrating controller, on the above, were both implemented on a basis of a SEU-hardened FPGA, including memory data integrated logic, so as to improve the reliability of spatial application.

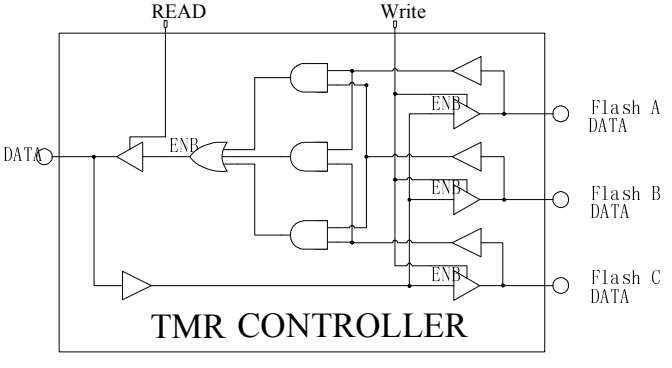


Fig. 6. The logic circuit of TMR of FLASH

Considering overseas import limit and the cost of SEU-hard product, we used SRAM-based FPGA based on COTS. There were several advantages, low-price, programmable, and widely-used, but it was easily incurred SEU. So, as bottleneck of system's reliability, SEU-mitigation design on FPGA was a sticking point of the system design.

3.3.1 The Inner Fault-Tolerant Design on FPGA

A. The redundant design on I/O pins

The signals of I/O pins were all triple-voted in the inner logic cells of FPGA. Then data signals and control signals could bear the effect of SEU while were being processed via inner logic of FPGA. As Shown in Fig.7, there was a SEU-tolerant design principle chart.

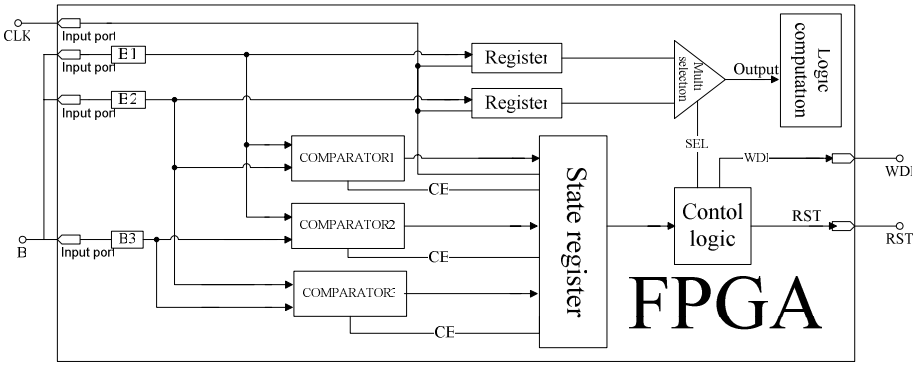


Fig. 7. The inner SEU-tolerant redundant architecture of FPGA

B. The dual-module redundant design on inner logic module

We have adopted the dual-module redundant design on the inner logic module M of FPGA. The thought was to integrate the advantages of high-speed hardware-based and economy time-based redundancy, which saved nearly 1/3 redundant hardware resource compared with traditional method TMR. The Scheme is shown in the fig.8 as follows: at the time of t, the input signal Input was exported to R(t) through the module M; then Input came by encoding and decoding to the redundant signal R(t+d). if the result compared was 1, which means the two signals reached the comparator C at the same time, R would be output, else returned a recalculation flag to reset the signal R from Input to M, and export the result to the signal Output.

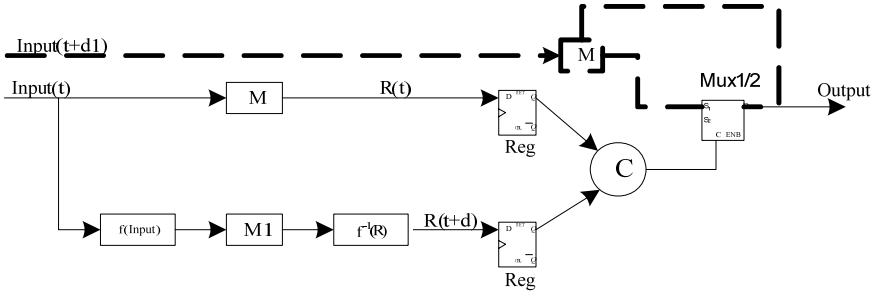


Fig. 8. The dual module redundancy design of the inner logic module of FPGA

C. The outer reconfigurable fault-tolerant design on FPGA

As the key to the implementation of inner hardware logic redundancy technique, the arbitrator was the bottleneck of the system reliability. So we have setup a watchdog (seen in the Fig.7) in the inner controller of FPGA. Once the watchdog sending an overtime signal, the logic result of the FPGA was wrong, and then it would resume from fault state through the online reconfiguration operation. However, in the course of deigning on computer system of spacecraft, we must implement its ability of auto-fault-repair. Thereby according to the characteristics of configuration of SRAM-based FPGA, we have designed an auto-reconfiguration circuit with watchdog. The typical configuration-circuit signal elements chart of SRAM-based FPGA was shown in the Fig.9, and the auto-repair function was triggered by the signal CONF_DONE.

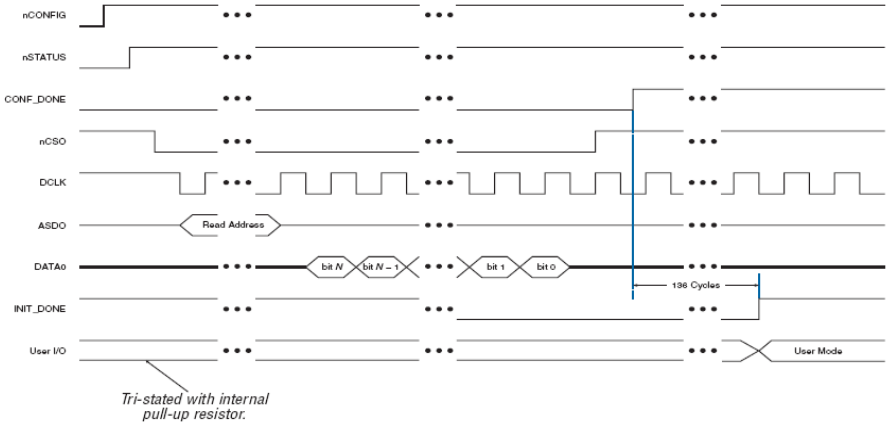


Fig. 9. The typical configuration-circuit signal elements chart of SRAM-based FPGA. When watchdog had alarmed, the signal CONF_DONE would be set to low level to reset the system.

The signal RST, the timeout alarm signal of watchdog, was connected with CONF_DONE. After the FPGA had been powered on and been configured, CONF_DONE became high level from low. Once the watchdog alarmed, RST would set CONF_DONE to low level, then the circuit of FPGA would restarted, and the inner logic circuits would be downloaded and auto-reconfigured from storage.

4 Analysis of the Reliability

The entire spacecraft computer subsystem was nonobjectively seen as a mixed combination system with serial and parallel connection. Seen in the Fig.10, the dual system came differently by fault-tolerant reinforcing to be the dual fault-tolerant system with parallel connection.

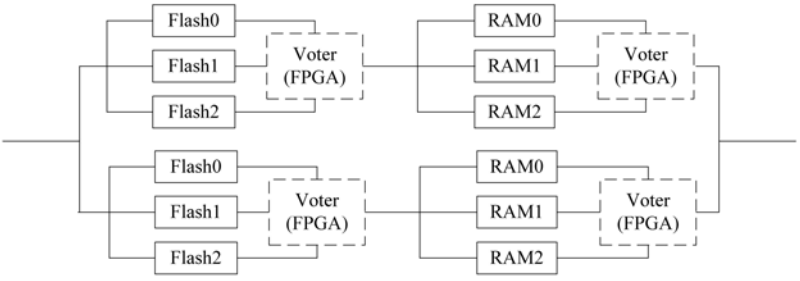


Fig. 10. The model on the reliability of SEU of OBC

4.1 The Reliability of a Single Board

Shown in the Fig.10, the single board was also the mixed combination model. The SEU-tolerant reliability of its modules (concluding Flash and RAM) was represented by Rm (Rm=R1R2), thereinto R1 denoted the reliability of Flash (or RAM) and R2 denoted the arbitrator of FPGA. λ denoted the SEU-invalidation effects, Rf and Rr respectly denoted the reliability of triple-module-redundant system of Flash and RAM, and the reliability of CPU in a single board was denoted by Rcpu. In the thesis, we have supposed that there were not more than one-bit SEU-upset in synchronization, and the invalidation effects of each module in the parallel connection modules was equal.

From the foregone conditions, results were obtained as follows.

$$R_{f10}=R_{f11}=R_{f12}=R_2=e^{-\lambda t}. \tag{1}$$

$$R_1=(e^{-\lambda t})^3+3(1-e^{-\lambda t})(e^{-\lambda t})^2=3e^{-2\lambda t}(1-e^{-\lambda t}). \tag{2}$$

$$R_m=R_1R_2=3e^{-3\lambda t}(1-e^{-\lambda t}). \tag{3}$$

$$R_f=R_r=R_m. \tag{4}$$

So the reliability of a single board could be denoted by

$$R_{cpu}=R_m{}^2=9e^{-6\lambda t}(1-e^{-\lambda t})^2. \tag{5}$$

4.2 The Reliability of the Dual System

From the Fig.10, the host and the spare made up of a mixed combination model through the judge controller of FPGA. The SEU-tolerant reliability of this system was denoted by Rd (Rd=Rcpu2). The equations were obtained as follows, according to the same hypothesis and known conditions presented before.

$$R_{cpu0}=R_{cpu1}=R_fR_r=R_m{}^2. \tag{6}$$

So the reliability of entire system could be denoted by

$$R=1-(1-R_m{}^2)^2=1-(1-9e^{-6\lambda t}(1-e^{-\lambda t})^2)^2. \tag{7}$$

Table 1. The SEU-tolerant reliability of OBC

Time(year)	R_m	R
1	0.9802	0.9996
2	0.8976	0.9895
3	0.7712	0.8600

In case of the disable probability of a single device was 10-7, and then the SEU-tolerant reliability of spacecraft computer system in apiece time extent could be shown in the table 1.

According to the reference[5], the disabled probability of computer subsystem should be less than 0.05 in two years. The result of above shows that its reliability satisfied the request.

5 Conclusion

With consideration of the characteristics of COTS-based devices and radiation effects in outer space, this thesis illustrated the design on SEU-tolerant architecture of COTS-based OBC. The fault-tolerant technology of its subsystem must give attention to the volume & power efficiency of spacecraft and meet the needs of appropriate redundancy and adequate reliability and the commercial request for low cost and high performance of COTS. So we analyzed function and logic structure of the OBC firstly. Secondly, we put forward an embedded design scheme with SEU-tolerant architecture. Finally, its reliability has been validated in theory. In the future, we will carry out the system to be validated in the spatial environment.

References

1. Du, W., Tan, W.: Research on Realizationof ASIC in Chinese Spacecraft. Chinese Space Science and Technology 5 (2002)
2. Reorda, S., Paccagnella, A.: Analyzing SEU Effects in SRAM-based FPGAs. IOLTS2003: IEEE International On-Line Testing Symposium , 119–123 (2003)
3. Pratt, B., Johnson, E., Wirthlin, M., Caffrey, M., Morgan, K., Graham, P.: Improving FPGA Design Robustness with Partial TMR. In: MAPLD 2005, Washington, D.C. (September 2005)
4. Lai, A.: Mitigation techniques for electronics in Single Event Upset environments[EB/OL]. US: Opensystems Publishing, Military Embedded Systems, [2006-6-27]. (2006), http://www.mil-embeded.com/articles/authors/lai
5. Xiang, L., Qu, G.: The Fault Tolerant System Design of Housekeeping Computer for small satellite. Aerospace Control 2, 92–96 (2005)

Balancing Thread Partition for Efficiently Exploiting Speculative Thread-Level Parallelism

Yaobin Wang, Hong An, Bo Liang, Li Wang, Ming Cong, and Yongqing Ren

Department of Computer Science and Technology,
University of Science and Technology of China, Hefei 230026, China
Key Laboratory of Computer System and Architecture,
Chinese Academy of Sciences, Beijing 100086, China
wyb1982@mail.ustc.edu.cn, han@ustc.edu.cn,
{boliang,wangliii,mcong,renyq}@mail.ustc.edu.cn

Abstract. General-purpose computing is taking an irreversible step toward on-chip parallel architectures. One way to enhance the performance of chip multiprocessors is the use of thread-level speculation (TLS). Identifying the points where the speculative threads will be spawned becomes one of the critical issues of this kind of architectures. In this paper, a criterion for selecting the region to be speculatively executed is presented to identify potential sources of speculative parallelism in general-purpose programs. A dynamic profiling method has been provided to search a large space of TLS parallelization schemes and where parallelism was located within the application. We analyze key factors impacting speculative thread-level parallelism of SPEC CPU2000, evaluate whether a given application or parts of it are suitable for TLS technology, and study how to balance thread partition for efficiently exploiting speculative thread-level parallelism. It shows that the inter-thread data dependences are ubiquitous and the synchronization mechanism is necessary; Return value prediction and loop unrolling are important to improve performance. The information we got can be used to guide the thread partition of TLS.

1 Introduction

We have witnessed that chip multiprocessors (CMPs), or multi-core processors, have become a common way of reducing chip complexity and power consumption while maintaining high performance. General-purpose computing is taking an irreversible step toward on-chip parallel architectures [1]. The ability to place multiple cores or many cores on the same chip will significantly increase the communication bandwidth and decrease the communication latency seen by threads executing on different processing cores. This enables the exploitation of finer-grained thread-level parallelism on a multicore chip as compared to a conventional symmetric multiprocessor (SMPs). But current parallel software is limited since many programs have been written using serial algorithms.

On the one hand, software transformations are a possible way for extracting some parallelism from these codes. Unfortunately, although parallel compilers have

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 40–49, 2007.

made significant efforts, they still fail to automatically parallelize general-purpose single-threaded programs which have complex data dependence structures caused by non-linear subscripts, pointers, or function calls within code sections [2,3,4]. On the other hand, many applications may still turn out to have a large amount of parallelism, but are still only hand-parallelizeable with state-of-the-art parallel programming models. Manual parallelization can provide good performance, but typically requires not only a different initial program design but also programmers with additional skills and efforts. In a word, the primary problem is that creating parallelized versions of legacy code is difficult.

Can simple hardware support on multicore chip help to parallelize general-purpose programs? To parallelize these codes, researchers have proposed Thread-Level Speculation (TLS) that allows to parallelize regions of code in the presence of ambiguous data dependence, thus extracting parallelism whatever dynamic dependences actually exist at run-time [5,6,7]. Speculative CMPs use hardware to enforce dependence, allowing a parallelizing compiler to generate multithreaded code without needing to prove independence. In these systems, a sequential program is decomposed into threads to be executed in parallel; dependent threads cause performance degradation, but do not affect correctness. Speculative threads are thus not limited by the programmer's or the compiler's ability to find guaranteed parallel threads. Furthermore, speculative threads have the potential to outperform even perfect static parallelization by exploiting dynamic parallelism, unlike a multiprocessor which requires conservative synchronization to preserve correct program semantics. But for performance reasons, thread decomposition is expected to reduce the run-time overheads of data dependence, inter-thread control-flow misprediction, and load imbalance. Unfortunately, these kinds of threads are very hard to find, especially in non-numerical programs. Identifying the points where the speculative threads will be spawned becomes one of the critical issues of this kind of architectures.

Several hardware designs have been proposed for this speculative thread-level parallelism (STP) model [5,6,7,9,10], but so far the speedup achieved on large general-purpose code has been limited. The decision on where to speculate can make a large difference in the resulting performance. If the performance is poor, we gain little insight on why it does not work, or whether it is the parallelization scheme or machine model (or both) that should be improved. As a consequence, poor results may not reflect any inherent limitations of the STP model, but rather the way it was applied.

The goal of this paper is to propose a criterion for selecting the region to be speculatively executed and to identify potential sources of speculative parallelism in general-purpose programs. We also evaluate whether a given application or parts of it are suitable for TLS technology, and study how to balance thread partition for efficiently exploiting speculative thread-level parallelism.

The rest of this paper is organized as follows. In Section 2 we describe the STP models for subroutine and loop level speculation. The analysis method is described in Section 3, followed by experiment analysis in Section 4. Finally we conclude in Section 5.

2 Speculative Thread-Level Parallel Execution Model

2.1 Candidate Threads

The thread partition is based on the control flow information, usually choose loop and subroutine structures as the candidate threads. For subroutine, its boundaries often separate fairly independent computations, the local variables wouldn't violate with the outer program; and for the loop body, every iteration does the similar operations to the same data set, and is independent each other. The data dependence between iterations is regular. Both of them are good choices for candidate threads.

2.2 Speculative Execution Model for Loops

The speculative execution model for loops is shown in Fig.1, for comparing, Fig.1(a) shows the traditional execution model and Fig.1(b) shows the speculative execution model. At the beginning of the speculative execution, the main processor informs all the other processors to load and execute different iterations of the loop by sending a "Loop_Start" signal to them. In the process of speculative execution, only the head processor can write to memory directly, and all the other speculative processor's memory references will be cached in its speculative buffer. The next processor will become the new head processor after the current head processor committed. A new iteration will be loaded and executed after a processor committed its result into memory. When a processor found that the exit condition of the loop becomes true, a "Loop_End" signal would be send to all the other processors to finish the speculative execution of the specific loop structure, and only the main processor continue running the code followed the loop.

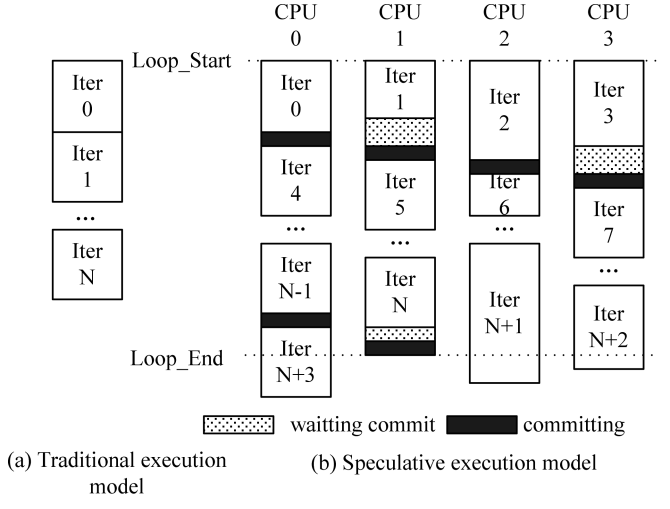


Fig. 1. Speculative execution model for loops

2.3 Speculative Execution Model for Subroutines

As shown in Fig.2, when a speculative subroutine call takes place, a new processor will be selected to run the code followed the call speculatively with the predicted return value and the old processor concurrently run the subroutine. The new processor's memory references will be cached in its speculative buffer while the old processor can write directly into memory. After the old processor complete the execution of the subroutine, the real return value come into being and compared with the prediction value, if miss prediction detected, the new processor must rollback to correct the execution.

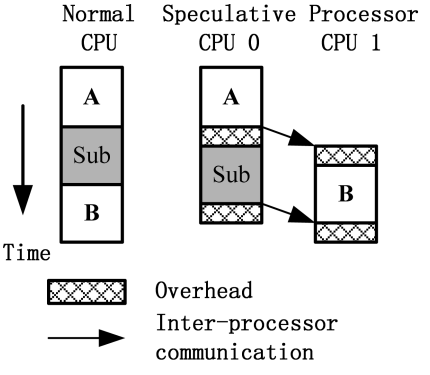


Fig. 2. Speculative execution model for subroutines

3 Analysis Method

3.1 Basic Criterion for Selecting Threads

Granularity and inter-thread data dependence pattern is the most important criterions for selecting candidate threads. Long thread may lead to speculative buffer overflow which must stall the execution of the thread, while short thread cannot payoff the overhead of speculative execution. Different from subroutine, loop slicing and unrolling can be used to control the granularity of a loop. Inter-thread data dependence pattern is the other basic criterion for both loop and subroutine, and we will propose two new concepts to describe this issue in section 3.2. Besides granularity and inter-thread data dependence pattern, there are some other distinguished criterions, such as type of return value and return value prediction rate that should be used to choose thread from subroutine structure. In all of them, value predication rate, data dependence, thread granularity are foremost in the TLS parallelism, the reasons are as follows:

The value predication rate shows the control dependence violations among the threads, the data dependence would cause the violations, and the granularity shows the problem about the thread balance.

3.2 Analysis Method for TLS Parallelism

The inter-thread data dependence can be abstracted as a producer/consumer model, write operation is data producing while read operation is data consuming. To describe the data dependence violation, we introduce two terms here: "produce- distance" and "consume-distance", as shown in Fig.3. The produce-distance means the instruction numbers from the beginning of the thread to the last write instruction for a specific memory address, and consume-distance means the instruction numbers from the beginning of the thread to the first read instruction for a specific memory address. Either of them is a concept relative to program's one specific execution and both of them must be calculated at running time.

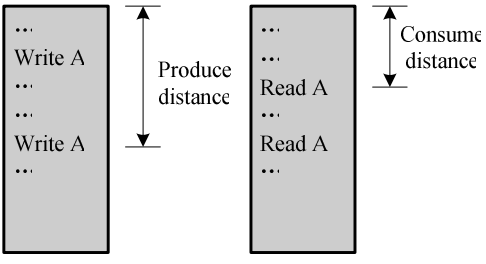


Fig. 3. Produce-distance & Consume-distance

For thread i and its successor thread i+1, starting at almost the same time, if the latter's consume-distance is less than the former's produce-distance, there will be a dependence violation under the assumption of that all processor execute instructions at a same speed. In this paper we select the ratio of consume-distance to produce-distance to evaluate the inter-thread data dependence pattern. To facilitate the description, we call a inter-thread data dependence as a Deadly Dependence if the ratio is less than 1.0, Dangerous Dependence for the ratio between 1.0 and 2.0 and Safe Dependence for the ratio larger than 2.0.

4 Experiment Analysis

4.1 Experimental Environment and Tools

The profiling tools we used in our investigation named ProFun, ProRV and ProLoop, and all of them are extended from sim-fast, the fasted simulator of SimpleScalar tool set which execute one instruction per cycle. ProFun and ProRV are used to profile the subroutines and ProLoop is for Loop. All the tests were achieved on an x86 machine running Linux system, the compiler we used is modified from gcc-2.7.2.3, and the benchmarks are selected from SPEC CPU2000.

Firstly, we pick out subroutines that occupy more than 5% of the total program execution time, as the inputs of ProRV and ProFun by using Gprof tool, and the input

loop structures of ProLoop were selected in the subroutines acquired above. And then by running the profiling tools, we profiled execution time distribution and the return value prediction rate of subroutines with different return value types, the granularity distribution and the inter-thread data dependence pattern of both subroutines and loops, and the ideal speedup achieved by speculative execution.

4.2 Experiment Results

4.2.1 Return Value Prediction Rate

The "sparse int" subroutine always returns zero except it errors and can be well predicted (e.g. boolean type), it is necessary to separate it from int type and we named it as sparse int. As shown in Fig.4, we found that the last-value prediction scheme is better then the stride, the "sparse int" type achieve a prediction rate about 80%, and the prediction rate of float is almost zero. For the "void" and "sparse int" subroutines take up most of the execution time and they're easy to predicate, we can say that the source of speculative thread-level parallelism is abundant in general-purpose applications.

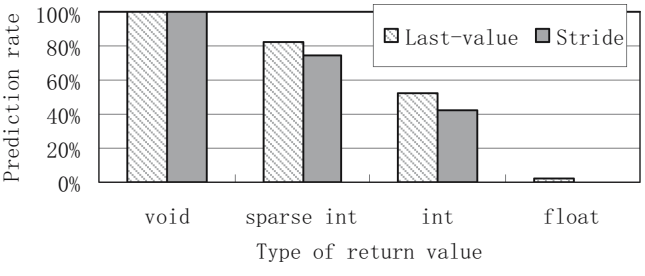


Fig. 4. Prediction rate of different return value types

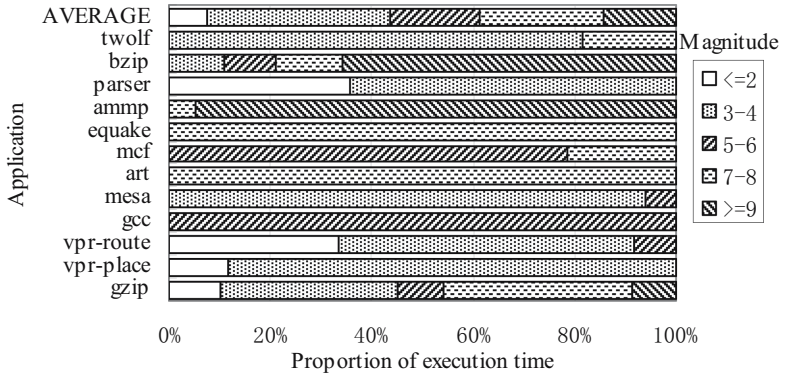


Fig. 5. The thread granularity for subroutines

Fig. 6. The thread granularity for loops

4.2.2 The Thread Granularity

Figure 5 shows the execution time distribution of subroutines with different instruction numbers, and from it we can see that the execution time distribution of them varied widely. For only the subroutines with a length of 10^3-10^6 instructions are suitable for speculative execution and luckily we found that they take about 55% of total execution time.

Figure 6 shows the execution time distribution for Loop structure, and we found that most of the loop iteration is shorter than 10^4 instructions. It means that loop unrolling should be frequently used to achieve a larger iteration.

4.2.3 Memory Dependence Distribution

Figure 7 and Fig.8 show the memory dependence distribution for subroutine and loop structures. From Fig.7 we can see that the distribution characteristic is quite varied,

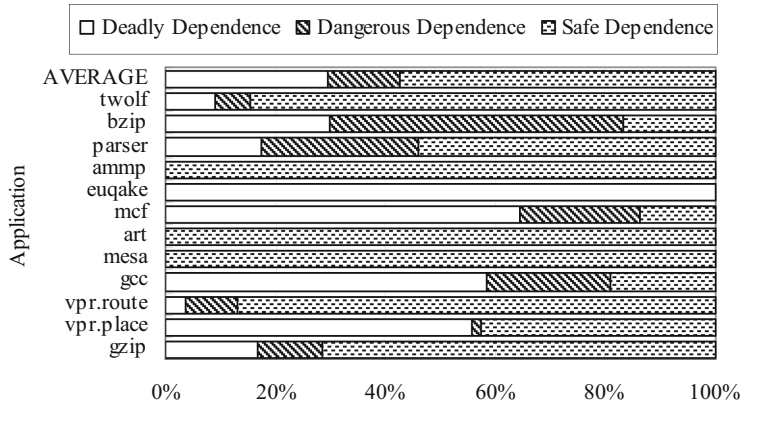


Fig. 7. Memory dependence distribution for subroutine speculation

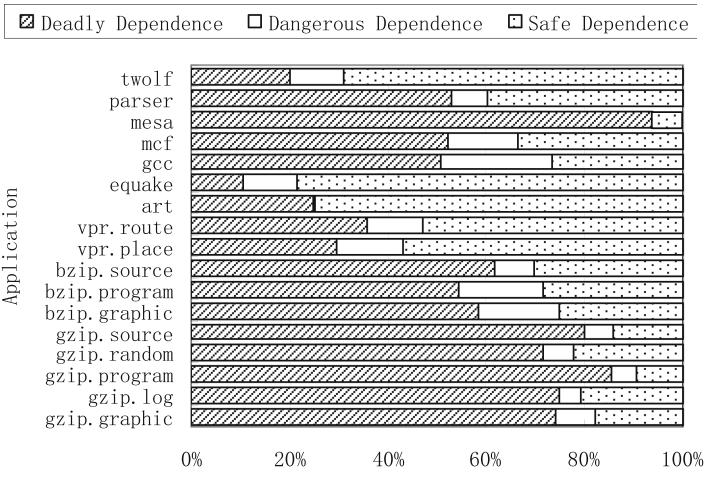


Fig. 8. Memory dependence distribution for loop speculation

and in average, there are about 30% dependences are deadly dependence, 12% are dangerous dependence and about 58% are safe dependence. Only ammp, art and mesa almost have no deadly dependence. The situation for loop structure is more pessimistic, all the applications have deadly dependence as shown in Fig.8. It means that for application of SPEC CPU2000, the data dependences are ubiquitous; to achieve a better performance, synchronization mechanism is quite necessary.

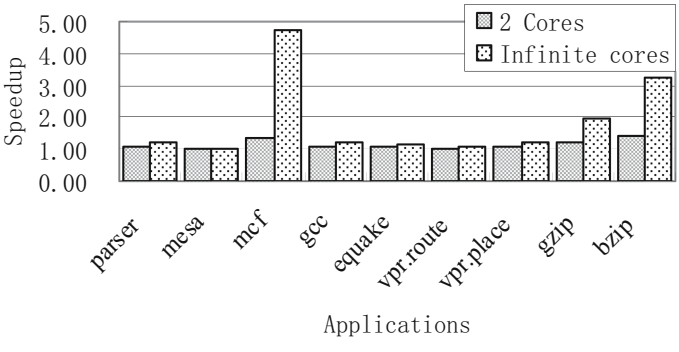


Fig. 9. Speedup of subroutine speculation

4.2.4 Speedup

Figure 9 and Fig.10 show the potential speedup of speculative execution for subroutine and loop structures when using different core numbers. As we can see in Fig.9, even the highest speedup we can achieve by speculative execution of subroutines using infinite cores, can not exceed 5, and most of the applications can only achieve the speedup lower than 3 even using infinite cores. The situation is more awful when limited the number of cores to 2. The similar situation was also appeared in Fig.10: the highest speedup can only be about 5.3 even for infinite cores and allowing speedup nest loop. This pessimistic result is consistent with the analysis mentioned in section 4.2.3.

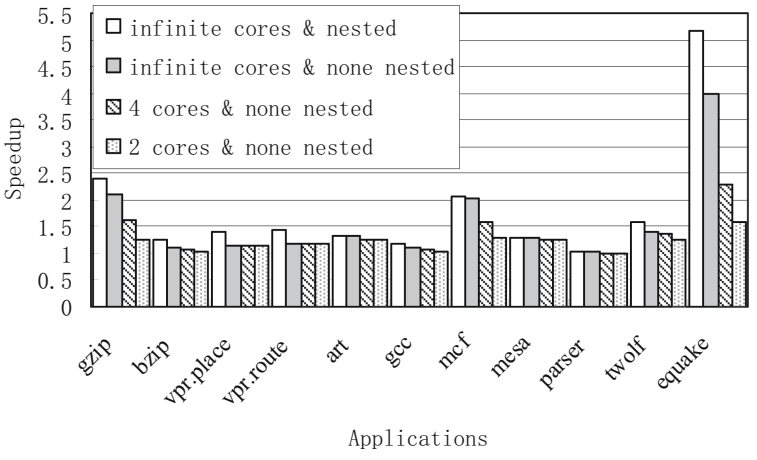


Fig. 10. Speedup of loop speculation

5 Conclusions

In this paper, a criterion for selecting the region to be speculatively executed is presented to identify potential sources of speculative parallelism in general-purpose programs. The dynamic profiling method has been provided to search a large space of parallelization schemes. We analyze the key factors impacting speculative thread-level parallelism of SPEC CPU2000, such as the return value prediction rate of subroutines with various prediction methods, the memory dependence, the granularity of loops and subroutines, and so on. We evaluate whether a given application or parts of it are suitable for TLS technology, and study how to balance thread partition for exploiting speculative thread-level parallelism. It shows that the source of speculative thread-level parallelism is abundant in general-purpose applications, the value predication and loop unrolling technology can greatly improve the TLS performance.

Acknowledgement

This work has been supported by the grant from Intel (PO#4507176412), the National Natural Science Foundation of China (60373043 and 60633040) and the National Basic Research Program of China (2005CB321601).

References

1. Asanovic, K., Bodik, R., et al.: The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report, No.UCB/EECS-2006-183, UC Berkeley (2006)
2. Zhai, A., Colohan, C.B., Steffan, J.G., et al.: Compiler optimization of scalar value communication between speculative threads. In: ASPLOS-10, San Jose, California (2002)
3. S.W. Liao, et al.: SUIF Explorer: An Interactive and Interprocedural Parallelizer. In: PPoPP 1999 (1999)

4. Miller, B.P., et al.: The Paradyn Parallel Performance Measurement Tools. IEEE Computer 11, 37–46 (1995)
5. Sohi, G.S., Breach, S.E., Vijaykumar, T.N.: Multiscalar Processors. In: 22nd Annual International Symposium (1995)
6. Hammond, L., Willey, M., Olukotun, K.: Data Speculation Support for a Chip Multiprocessor. In: ASPLOS-VIII, San Jose, CA (1998)
7. Steffan, J.G., Mowry, T.: The potential for using thread-level data speculation to facilitate automatic parallelization. In: HPCA-4, Las Vegas, NV (1998)
8. Oplinger, J.T., Heine, D.L.: In Search of Speculative Thread-Level Parallelism. In: Malyshkin, V. (ed.) Parallel Computing Technologies. LNCS, vol. 1662, Springer, Heidelberg (1999)
9. Akkary, H., Driscoll, M.A.: A Dynamic Multithreading Processor. MICRO-31, Dallas, TX (1998)
10. Krishnan, V., et al.: Hardware and Software Support for Speculative Execution of Sequential Binaries on a Chip- Multiprocessor. In: Supercomputing 1998, Melbourne, Australia (1998)

Design and Implementation of a High-Speed Reconfigurable Modular Arithmetic Unit

Wei Li, Zibin Dai, Tao Chen, Tao Meng, and Xuan Yang

Institute of Electronic Technology, the PLA Information Engineering University, Zhengzhou 450004, China
try_1118@163.com

Abstract. A high-performance and dynamic reconfigurable modular arithmetic unit is presented, which provides full support to modulo $2^8/2^{16}/2^{32}$ addition and modulo$2^{32}/ 2^{16}+1/2^{32}$-1 multiplication operation. To save the hardware cost, we have adopted sharing technique to implement modular multiplication operation, and then optimized each critical block. The design has been realized using Altera's FPGA. Synthesis, placement and routing of reconfigurable design have accomplished on 0.18μm SMIC process. The result proves that the propagation time of the critical path is 6.04ns. Compared with other designs, the reconfigurable modular arithmetic unit not only supports for diverse modular arithmetic in the block ciphers, but also provides IP Core for reconfigurable cryptographic system.

1 Introduction

Block ciphers are the core mechanism of data encryption, digital signature and key management due to the characteristic of high-speed, simplified standardization and hardware realization. With the development of cipher chip design technology, the realization method of cipher algorithms is increasing gradually. The cipher processor is widely approved due to the characteristic of high-speed and flexible realization. The design scheme of reconfigurable cipher processing unit is: the hardware circuit can be reconfigured to adapt to more cipher algorithms and have a tradeoff between flexibility and efficiency.

Based on the analysis of block cipher architectures, most designs of block cipher algorithms have a few similar hardware units. In the block cipher algorithms, the frequency of modular arithmetic operation is very high. As for different block ciphers, there are large difference in the aspects of width and module. So the modular arithmetic unit not only takes up large area but also pays more delay time. Basing on the analysis of modular arithmetic operation, this paper presents a high-speed reconfigurable modular arithmetic unit(RMAU), which can be reconfigured to perform modulo $2^8/2^{16}/2^{32}$ addition and modulo$2^{32}/ 2^{16}+1/2^{32}$-1 multiplication operation.

This paper is organized as follows: section 2 presents a reconfigurable modular arithmetic unit. Section 3 optimizes the 16-bit multiplier with booth algorithm[1], leapfrog Wallace tree[2] and Ling adder[3]. Section 4 analyzes conventional adders in delay and area. Section 5 introduces a high-speed module modification circuit.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 50–59, 2007.

Section 6 shows the simulation result and compares the RMAU with reference[4],[5]. Finally, we conclude with section 7.

2 Design and Application of RMAU Unit

2.1 Analysis of Modular Operation Based on Block Ciphers

In the block cipher algorithms, the frequency of modular operation is very high. However, different algorithms have different operation mode and width. Based on the analysis of conventional block ciphers, if the width of two operands is n, module is usually 2^n or $2^n \pm 1$, and n is 8, 16 or 32. Table 1 shows the modular operation of published block cipher algorithms.

Table 1. Modular operation of block cipher

Algorithm	Addition/Subtraction	Multiplication
IDEA	Modulo 2^{16}	Modulo2^{16}+1
Blowfish	Modulo 2^{32}	
MARS	Modulo 2^{32}	Modulo2^{32}
FEAL	Modulo 2^8	
CAST-128	Modulo 2^{32}	
CAST-256	Modulo 2^{32}	
MMB		Modulo2^{32}-1
GOST	Modulo 2^{32}	
WAKE	Modulo 2^{32}	
WiderWake	Modulo 2^{32}	
RC2	Modulo 2^{16}	
RC5	Modulo 2^{32}	
RC6	Modulo 2^{32}	Modulo2^{32}
SAFER K	Modulo 2^8	
SAFER +	Modulo 2^8	
E2		Modulo2^{32}
Twofish	Modulo 2^{32}	

Based on the analysis above, conventional modular operation includes modulo 2^{16} addition or subtraction, modulo 2^{32} addition or subtraction, modulo 2^{16}+1 multiplication and modulo 2^{32}-1 multiplication. Therefore, the configurable modular operation unit should support to implement the operation above through dynamically reconfiguring the control signals.

2.2 Hardware Architecture of RMAU

The whole architecture of RMAU is illustrated in Fig.1. It contains 16-bit multiplier, adder array block, modulo 2^{16}+1 and 2^{32}-1 modification circuit and some reconfigurable control signals (C1~C5). Beside that, the function of invers block is that it can acquire the negative logic of data input under the domination of signal C3.

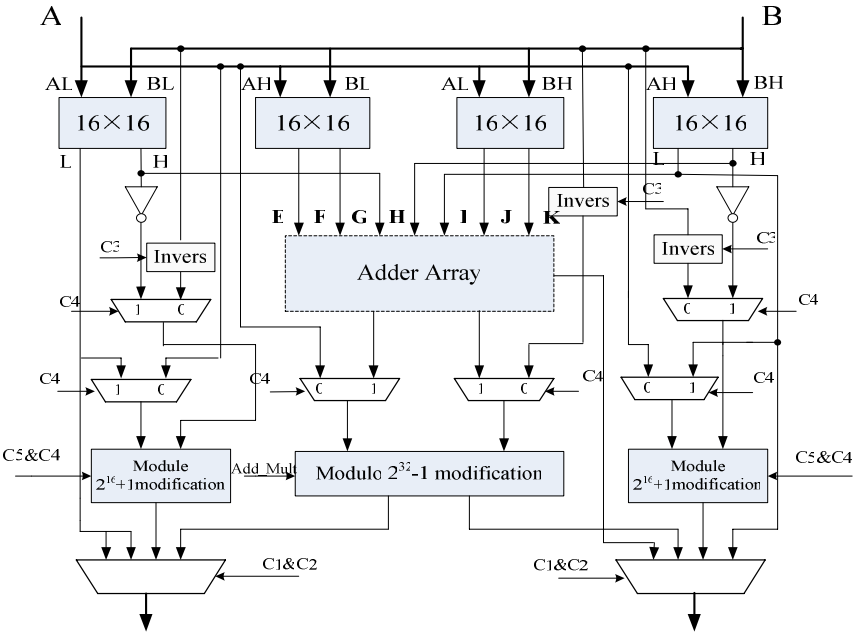


Fig. 1. The whole architecture of RMAU

Table 2. The details of C1~5's configurations

Mode	Control signal				Operation
	C1&C2	C3	C4	C5	
Config1	00	X	X	X	Modulo 2^{16} multiplication
Config2	01	X	X	X	Modulo 2^{32} multiplication
Config3	10	0	0	0	Modulo 2^{8} addition
Config4		1	0	0	Modulo 2^{8} subtraction
Config5		0	0	1	Modulo 2^{16} addition
Config6		1	0	1	Modulo 2^{16} subtraction
Config7		X	1	X	Modulo 2^{16}+1 multiplication
Config8	11	0	0	X	Modulo 2^{32} addition
Config9		1	0	X	Modulo 2^{32} subtraction
Config10		X	1	X	Modulo 2^{32}-1 multiplication

In the architecture of RMAU, it contains many reconfigurable elements whose function is determined through programmable configuration bits such as C1~C5. These elements are connected using a set of routing resources that are also programmable. In this way we can change the configurable routing to connect the blocks together to form necessary circuit. At the same time, we can save hardware sources. In this paper, through reconfiguring control signals (C1~C5), the RMAU can implement modular addition, modular subtraction and modular multiplication operation.

Moreover, it can also implement modulo $2^{16}+1$ and modulo $2^{32}-1$ multiplication operation. The details of C1~C5's configurations are presented in Table 2.

2.3 Design Scheme of RMAU

In the architecture of RMAU, 16-bit multiplier, 16-bit adder, 32-bit adder and module modification circuit are the basic blocks of RMAU. So RMAU can implement modular addition and modular subtraction operation by setting the signals C1~C5. Moreover, because it has 16-bit multiplier, RMAU can also implement modulo 2^8 and modulo 2^{16} multiplication operation. The illustration about modulo $2^{16}+1$, modulo 2^{32} and modulo $2^{32}-1$ multiplication scheme will be given as follows.

Modulo 2^{32} multiplication operation can be expressed as $A\times B \bmod 2^{32}$, where A and B are 32-bit data. AH and BH denote 16 most significant bits of A and B. AL and BL denote16 least significant bits of A and B.

(i) $A\times B=C_1 2^{32}+C_2$, where $0\le C_1, C_2<2^{32}$
(ii) $AL\times BL=D_1 2^{16}+D_2$, $AL\times BH=E_1 2^{16}+E_2$
(iii) $AH\times BL=F_1 2^{16}+F_2$, where $0\le D_1, D_2, E_1, E_2, F_1, F_2<2^{16}$

Therefore, we may get

$C_2[31:16]=(D_1+E_2+F_2)\bmod 2^{16}$, and $C_2[15:0]=D_2$

From the equation above, C_2 is the result of $A\times B \bmod 2^{32}$

As to modulo $2^{16}+1$ multiplication operation, we design the specific module modification circuit. There are two main realization methods: One is Ma algorithm[6], which can modify directly in the process of multiplication operation. Another method is Low—High algorithm scheme, which can modify the result after multiplication operation. In this paper, we adopt Low—High algorithm to realize module modification circuit. The modified scheme of modulo $2^{16}+1$ multiplication operation is:

$A\times B=(C_1 2^{16}+C_2)\bmod (2^{16}+1)$, where $0\le C_1, C_2<2^{16}$
Since $C_1 2^{16}+C_2=C_1(2^{16}+1)+(C_2-C_1)$
Therefore, $(C_1 2^{16}+C_2)\bmod (2^{16}+1)=(C_2-C_1)(2^{16}+1)$

From the equation above, C_2-C_1 is the result of $A\times B \bmod 2^{32}$

As to modulo $2^{32}-1$ multiplication operation, it can be easily seen that the modified scheme is the same as modulo $2^{16}+1$ multiplication. We adopt Low+High algorithm to modified 32-bit multiplication result. The scheme is described as follows.

$A\times B=C_1 2^{32}+C_2$, where $0\le C_1, C_2<2^{32}$
Where,$(AL\times BL=D_1 2^{16}+D_2,\ AL\times BH=E_1 2^{16}+E_2,\ AH\times BL=F_1 2^{16}+F_2,$
$AH\times BH=G_1 2^{16}+G_2)$

Here $((0\le D_1, D_2, E_1, E_2, F_1, F_2, G_1, G_2)<2^{16})$
We may get $C_1 2^{32}+C_2=C_1(2^{32}-1)+(C_2+C_1)$
Since $C_2[15:0]=D_2$, $C_2[31:16]=(D_1+E_2+F_2)\bmod 2^{16}$

$C_1[15:0]=\{((D_1+E_2+F_2)\bmod 2^{16})\text{cout}+(E_1+F_1+G_2)\bmod 2^{16}\}\bmod 2^{16}$
$C_1[31:16]=\{(((D_1+E_2+F_2)\bmod 2^{16})\text{cout}+(E_1+F_1+G_2)\bmod 2^{16})\bmod 2^{16})\text{cout}+G_1\}\bmod 2^{16}$
(Here, $(X\bmod 2^{16})$cout is the carry of $X \bmod 2^{16}$)

Based on the analysis above, it can be seen that the series carry increase the delay time of whole path. In order to minimize the delay time of critical path, we incorporate the carry together. Therefore, the equation can be described as follows.

$C_1=(D_1+E_2+F_2)\bmod 2^{16})\text{cout}+((E_1+F_1+G_2)\bmod 2^{16})\text{cout}2^{16}+(E_1+F_1+G_2)\bmod 2^{16}+G_1 2^{16}$
we can obtain: $(C_2+C_1)\bmod(2^{32}-1)$ is the result of $A\times B \bmod(2^{32}-1)$

3 Analysis and Realization of 16-Bit Multiplier

As the whole circuit architecture, 16-bit multiplier is the core block, but it has a long delay time. So we make the optimization to improve its performances in speed, area and power. The improved 16-bit multiplier we designed is shown in Fig.2. We consider the optimization of multiplier mainly from three parts: Firstly, through compared with other booth algorithms, we adopt modified booth 2 algorithm[7], Secondly, the traditional Wallace tree is replaced by leapfrog Wallace tree which minimizes the delay time of partial products compression. Finally, we adopt the Ling adder[3] to get the final result. The detailed illustration about Ling adder is depicted in 4.1.

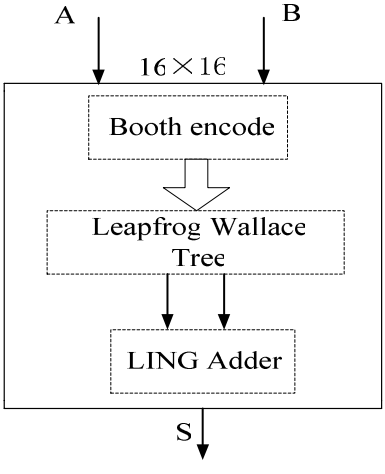


Fig. 2. Structure of 16-bit multiplier

3.1 Analysis of Booth Algorithm

Recently, a recoding scheme introduced by Booth [8] reduces the number of partial products by about a factor of two. But there is different characteristic in various booth algorithms. The analysis result is shown in Table 3.

Table 3. Analysis of booth algorithms

Algorithm type	Partial products number	Required extra circuit
None	16	None
Booth3	6	Shifting, inverting, 3M.
Booth4	5	Shifting, inverting, 3M, 5M, 6M, 7M.
Redundant Booth3	7 constant+2 carry =9	Shifting, inverting, 3M.
Modified Booth2	9	Shifting, inverting.

Based on the analysis above, booth algorithm has an obvious advantage in partial products compression. As shown in Table 3, booth 4 algorithm produces 4 partial products. But it requires more multiple generation circuit including 3M, 5M, 6M and 7M; Redundant booth 3 algorithm produces 6 partial products, and it requires 1 bias constant and two carry combining numbers. Moreover, it also requires multiple generation circuit; booth 3 algorithm products 6 partial products, and requires 3M generation circuit.

Modified booth 2 algorithm produces 9 partial products. But all of the multiples can be produced using simple shifting and complementing. So the modified booth 2 algorithm shows advantage in speed and area compared to other booth algorithms.

3.2 Analysis of Leapfrog Wallace Tree

As we know, Wallace tree architecture can enhance the compressing speed of partial products. The traditional Wallace tree architecture is shown in Fig.3.(a). Detailed illustration is shown in references [9]. In this paper, we have adopted the leapfrog Wallace tree (as shown in Fig.3.(b))which has a high performance in speed. Furthermore, it reduces the spurious switching in the circuit to save power dissipation.

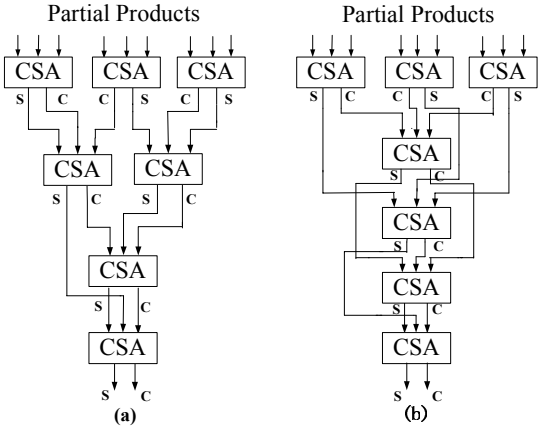


Fig. 3. Structure of Wallace tree

An improved method is proposed through adopting leapfrog Wallace tree architecture. The main idea of it is to make C and S signals enter next level CSA circuit separately. They can arrive at input port of every level CSA circuit at the same time. So the invalid inversion can be decreased. Compared with traditional Wallace tree architecture, there is not data waiting time in the leapfrog Wallace tree architecture, which accelerate compression of partial products.

4 Design of Adder Array Hardware Architecture

4.1 Analysis of Adder Unit

In the reconfigurable multiplier, 16-bit multiplier, module modification circuit and adder array all include the adder circuit. So the fast adder is important to high

performance multiplier design. Compared with conventional adder, we adopt Ling adder. In the Ling scheme, the group carry generate and propagate (G and P) are replaced by similar functions (called H and I respectively) which can be produced in fewer stages than the group G and P. we make the comparison between Ling adder and other conventional adders in area and speed performance. Synthesize adders with Synopsys Design Compiler, the result is shown in Table 4.

Table 4. Analysis of conventional adder

Adder type	Delay ns	Area μm^2
CPA	1.21	4896
BK	1.27	5121
KS	1.13	6433
CLA	1.19	5255
Synopsys Adder	1.19	3998
LING	1.07	5264

Based on the analysis above, Ling adder shows an obvious advantage in speed. The area of Ling adder is similar with BK[10], CPA and Synopsys adder. What is more, the size of Ling adder is smaller than KS[11] and CLA adder. So Ling adder is best choice in the consideration of speed and area.

4.2 Hardware Architecture of Adder Array

In the architecture of RMAU, adder array is the basic block which is shown in Fig.4. We optimize the block with CSA and LING adder. Beside, there are two combination elements in adder array. The function of it is to incorporate two input ports into one output ports. The adder array is designed to accomplish a series of addition operation. We put the result of adder array as inputs into the module modification circuit, and then accomplish modulo $2^{16}+1$ or $2^{32}-1$ multiplication.

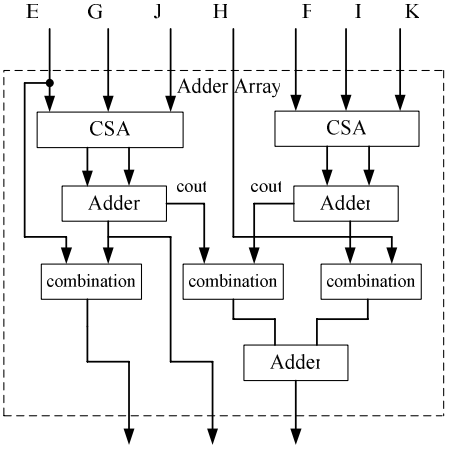


Fig. 4. Structure of adder array

5 Analysis and Implementation of Module Modification Circuit

The scheme about modulo 2^n+1 multiplication is illustrated in 2.3. We adopt Low-High algorithm. The architecture of module modification circuit is shown in Fig.5. The architecture not only provides support for modulo 2^n+1 multiplication implementing, but also accomplishes 8 or 16-bit modular addition and subtraction operation. In the process of modular multiplication operation, we segment the sum of 16-bit multiplier into n least significant bits (A) and n most significant bits (B), and then make them as inputs into the module modification circuit, which accomplishes A-B operation. As to modular addition and subtraction operation, we use signal C5 to distinguish between those two operations.

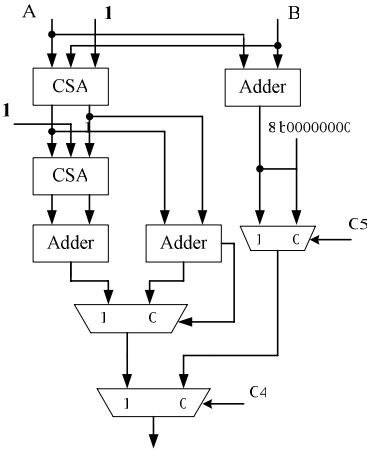


Fig. 5. Architecture of module modification circuit

6 Analysis and Comparison of Performance

6.1 Performance Evaluation of RMAU

Synthesize the reconfigurable modular arithmetic unit with Synopsys Design Compiler, based on SMIC 0.18μm CMOS technology, without regard to wire load, at worst case, the result in area and speed of RMAU are listed in Table 5

Slack indicates whether synthesis result meets the constraint. A positive Slack value denotes the design meets the required setup-time. And a negative Slack implies that the hold time of the endpoint flop is violated. As to our design, when the constraint is 8ns, the value of Slack is 0.00ns. Therefore, the maximum frequency is 125MHz.

Table 5. Performance of RMAU in area and speed

Constraint(ns)	Area(μm^2)	Slack(ns)
5	157480	-5.34
8	99697	+0.00
10	95127	+0.04

6.2 Contrast with Other Designs

In this paper, we compare RMAU with the reconfigurable elements of COBRA [4] and RELOG_DIGG[5] cipher processor. COBRA cipher processor designed special block to implement modulo 2^{32} multiplication and modulo 2^{32} addition operation. RELOG_DIGG cipher processor also designed special block to accomplish modulo $2^{16}+1$, 2^{32} multiplication and modulo 2^{32} addition operation. But in our design, RMAU can implement modular addition, modular subtraction and modular multiplication operation. Moreover, it can also implement modulo $2^{16}+1$ and modulo $2^{32}-1$ multiplication operation. RMAU has an obvious advantage in flexibility. The comparison in delay time between RMAU and special modular arithmetic element of two cipher processor are listed in Table 6. From these we can see: RMAU also has a high performance with other designs.

Table 6. The comparison with other designs

Design	Modular multiplication			Modular addition
	Modulo 2^{32}	Modulo $2^{16}+1$	Modulo $2^{32}-1$	Modulo 2^{32}
RELOG_DIGG	13.685ns	14.701ns	--	6.910ns
COBRA	5.5ns	--	--	5.0ns
RMAU	3.6ns	3.7ns	6.04ns	2.82ns

7 Conclusion

In this paper, we present a high-speed RMAU, which can achieve both short critical path and low power dissipation. Furthermore, we optimize 16-bit multiplier, module modification block and analyze conventional adder. Synthesize design with Synopsys Design Compiler, Simulation results show achievements on both high speed and small size.

References

1. Booth, A.D.: A Signed Binary Multiplication Technique. Quarterly Journal of Mechanics and Applied Mathematics 4, 236–240 (1951)
2. Nan, Y.S., Chen, O.T.: Low-power multipliers by minimizing switching activities of partial products. In: IEEE International Symposium on Circuits and Systems[C]. IEEE Circuits and Systems Society, Arizona USA, pp. 93–96 (2002)
3. Ling, H.: High-Speed Binary Adder. IBM Journal of Research and Development 25, 156–166 (1981)
4. Ying jie Qu.: The Research and Design of the Reconfigurable Logic for Cryptography[D]. In: Beijing University of Technology, Beijing China (2002)
5. Elbirt, A.J.: Reconfigurable Computing For Symmetric-Key Algorithms[D] Massachusetts: Electrical and Computer Engineering Department University of Massachusetts Lowell (2002)
6. Yu tai Ma: A Simplified Architecture for Modulo (2^n + 1) Multiplication. IEEE Transactions on Computers 47 (1998)

7. MacSorley, O.L.: High-Speed Arithmetic in Binary Computers. Proceedings of the IRE 49, 67–91 (1961)
8. Weinberger, A., Smith, J.L.: A One-Microsecond. Adder Using One-Megacycle Circuitry. IRE Transactions on Electronic Computers, 65–73 (1956)
9. Wallace, C.S.: A Suggestion for a Fast Multiplier. IEEE Transactions on Electronic Computers, 14–17 (1964)
10. Brent, P.R., Kung, T.H.: A regular layout for parallel adders. IEEE Trans. Comput. 32, 260–264 (1982)
11. Kogge, P., Stone, H.: A parallel algorithm for the efficient solution. IEEE Trans. Comput. 22, 783–787 (1973)

Virtual Disk Monitor Based on Multi-core EFI*

Xizhe Zhang[1], Shensheng Zhang[1], and Zijian Deng[2]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
[2] Lab of Information Security and National Computing Grid, Southwest Jiaotong University, Chengdu, 610031, China
{zhangxizhe,sszhang}@sjtu.edu.cn, zijian.deng@gmail.com

Abstract. This paper presents a *novel* approach to build a real-time high performance virtual disk monitor based on multi-core EFI, which keeps file system's integrity and mobility simultaneously. This monitor supports both file-level and block-level protection for general FAT file system. Data image can be dynamically loaded from various devices such as disk drives and even from a USB flash drive. A prototype system has been designed and implemented. Experiments show great performance merits for this unique virtual disk monitor.

1 Introduction

Extensible Firmware Interface (EFI) [1] was designed to modernize firmware technology and move beyond the limitations of legacy Basic Input/Output System (BIOS). By using EFI, we can effectively manage the platform firmware.

Disk virtualization is widely used in modern operating system (OS) and system virtual machine [12] [13] [14]. It is very useful for implementing portable file system and surveillance file system. Virtual disk emulates physical disk on block-level transfer operations so that transient status and operation information naturally embedded in physical disk can easily be retrieved and logged.

In general purpose OS and system virtual machine environment, no matter how disk virtualization is implemented, performance and security requirements can not be fulfilled at the same time. When OS is compromised, file system may also be corrupted and data integrity may not be preserved. System virtual machine suffers from performance overhead because of system emulation.

In this paper, a Virtual Disk Monitor (VDM) based on multi-core EFI is presented. The monitor is able to balance performance and security requirements simultaneously. This approach also extends EFI's ability to support a secure computational environment for future multi-core platform.

Main contributions of this paper are:

- A real-time virtual disk monitor on FAT file system was designed and implemented

* This project is supported by Intel Corporation and SJTU under the Contracts No. 4507258277 and No. 4507255994.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 60–69, 2007.

- Files access rules can be dynamically assigned by EFI user
- Experiment results show up to 3 times speedup on write operation as compared to that of physical hard disk

The rest of this paper is organized as the following: Section 2 discusses related works published previously; Multi-core EFI architecture is introduced in section 3; Section 4 describes the prototype design of EFI virtual disk monitor; Experiment results are presented in section 5; Finally, section 6 concludes the VDM project and discusses future work.

2 Related Work

Authors of reference [4] provided an accurate disk drive model, which precisely emulates physical disk drives. A real-time integrity monitor on file-system is proposed in reference [7]. Authors pointed out seven security problems associated with file-system integrity. They used Xen [11] VMM to transfer file system operations from Domain-U to Domain-0. However, this method needs to modify kernel code and needs a trusted kernel on Domain-U. In reference [6], authors presented an file-aware block level intrusion detection storage system, which can convert block level operations to file level operations. This method requires a scanning of the entire file system before turning on the intrusion detection feature. In addition, file access control rules must be defined before mounting virtual disk.

In reference [5], authors comprehensively explained the opportunities and obstacles on virtual machine based security architecture. They suggested an OS Interface library to translate hardware status from guest virtual machine to virtual machine introspection based intrusion detection system (VMI-based IDS).

Encouraged by the above researches and taking a step forward, Virtual Disk Monitor (VDM) is made capable of translating hardware status of disk I/O into file operations in real time. VDM can protect virtual file system without the need of a trusted kernel in OS. In addition, the new VDM does not need to scan the entire file system before mounting disk image. Moreover, with new VDM, file access control rules can be dynamically changed.

3 Multi-core EFI Architecture

Extensible Firmware Interface was invented in 1999. It was traditionally used for booting Intel Itanium processor-based sever. This section introduces EFI and Multi-core EFI architecture.

3.1 EFI

As a specification, EFI defines a new model for the interface between operating systems and platform firmware. It also provides boot and runtime service calls

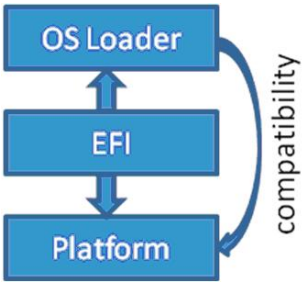


Fig. 1. EFI Architecture

available to the operating system and its loader. A diagram of EFI architecture is given in Figure 1.

Although primarily intended for the next generation of IA architecture-based computers, EFI is designed to be CPU architecture independent. It separates the hardware platform from the OS, but still let OS transparently control the platform. Under this condition, independently from the OS, EFI provides its own value-added features. EFI framework also specifies and provides a series of programmatic service interfaces to EFI applications and services, which are purely API specifications and implementation independent.

Intel created the platform Innovation Framework named Tiano for EFI. Tiano [3] made EFI extended for Desktop computers. Tiano can support unmodified general purpose OS like Windows XP, etc.

3.2 Multi-core EFI

Multi-core architecture is a single package that contains two or more processing "execution cores" or computational engines [2]. Accordingly, multi-core platform provides more processing resources.

In multi-core environment, EFI partitions hardware platform into many separate logical domains [9] [10]. EFI domain is a base and trusted domain. Other domains are un-trusted domains. Figure 2 illustrates this architecture. Bootstrap processor (BSP) is dedicated to EFI. Application processors (AP), physical devices and virtual devices can be assigned to the OS which is monitored by EFI. Although monitored OS has its own dedicated memory, EFI can access all physical memory address.

Monitored OS can be a general purpose OS such as Linux or Windows. EFI monitors the OS though the methods of memory inspection, system call monitoring and virtual device monitoring. Memory inspection method uses memory information of OS to interpret OS status. System call monitor method gathers OS information by insert a system call hooker into monitored OS. Virtual device monitor method transparently monitors OS through device operations and hardware status. This paper will focus on virtual device monitor method.

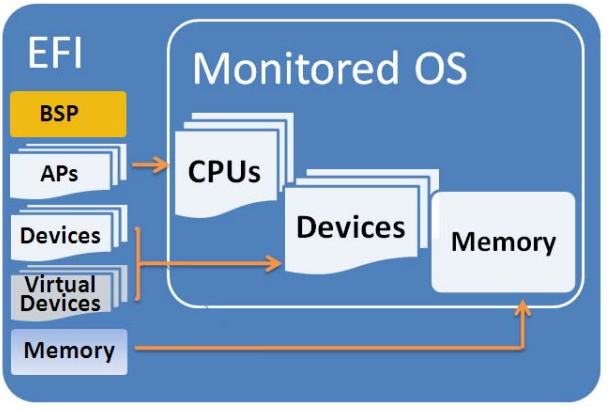


Fig. 2. Multi-core EFI Architecture

4 System Design

This section presents the design of Virtual Disk Monitor (VDM) in the following order: the structure of VDM and its components; program model; and the mechanism of VDM.

4.1 Structure

The design is based on multi-core EFI architecture. Linux operating system is used as monitored OS. For hardware, a dual-core processor is used as reference system platform. Figure 3 illustrates the structure of this virtual disk monitor system.

In figure 3, EFI is a host which has the access of all memory locations. Boot Strap Processor (BSP) is dedicated to EFI. Linux controls Application Processor (AP) and has its own dedicated memory. Both Linux and EFI can access shared memory used for communication between two systems.

On EFI site, the components of VDM consists of management console, disk image, access rules and Monitor. Users use management console to load disk image from various devices in EFI directory or even from a USB flash drive. After disk image is loaded, users can view files and directories in the disk image. Access rules can also be assigned and modified by users through the management console. Monitor checks access rules on every block write and read operation.

Management console is connected through a serial port. This serial port is concealed by EFI and not recognized by OS. To the OS, this serial port does not exist. The console is alive after the system powers on.

For general purpose OS, device driver is the only way to access various storage media. Virus or malicious programs can cause troubles to a system by corrupting OS or by corrupting device driver or by both. Since users assign access rules to files by OS interface, they may be fooled by corrupted OS. Further, OS may be

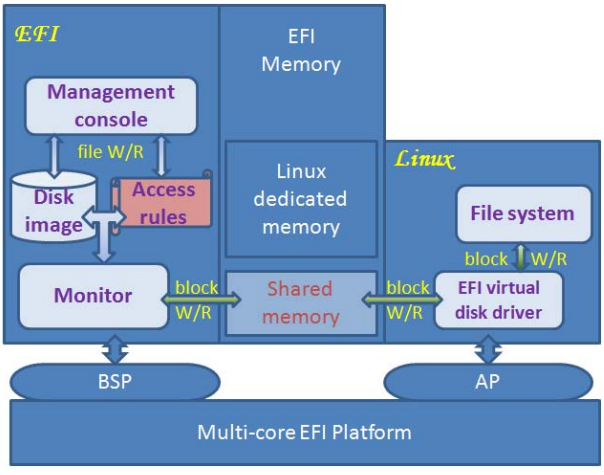


Fig. 3. Virtual Disk Monitor System Structure

fooled by corrupted device driver. The main issue here is that the storage media can not report the problems to users directly.

In multi-core EFI system, storage media is emulated by VDM. Device can not fool users by conceal or forge real media operations. Any violence of access rules bill be reported to users on EFI console immediately. Then user can unmount the virtual disk. EFI domain is a trusted base in the system. As long as EFI domain remains intact, un-trusted domain can not corrupt storage media controlled by VDM.

VDM check every disk I/O operations. VDM first check the access rules then perform real disk operation. Rules Checking Additional Time (RCAT) is different according to different type of access rule. For block level access rules, RCAT is a constant. For file level access rules, RCAT varies as the depth of directory where files reside varies.

When a user assigns access rule to a file, VDM marks relevant files blocks and relevant directory blocks according to the file. Mark Time (MT) will also vary as the depth of directory where the file resides varies.

On Linux site, EFI virtual disk driver is a loadable kernel module which acts as a block device. File system calls it when users write to or read from EFI virtual disk. EFI virtual disk driver will then relay the file system call information to the Monitor in EFI through shared memory.

Write and read operations among Linux file system, EFI virtual disk driver and EFI monitor are block level operations. EFI management console and Linux applications access virtual disk through file operations.

4.2 Program Model

In presented design, a USB flash drive is attached to EFI system. On software level, Linux applications see no difference between EFI virtual disk and physical disk although the underlying physical mechanism has been changed.

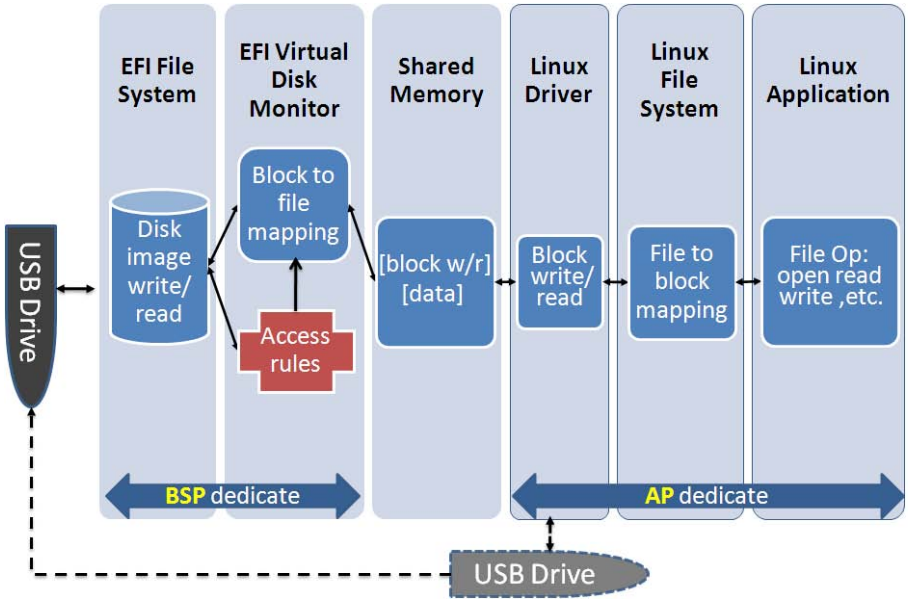


Fig. 4. Virtual Disk Monitor Program Model

Linux applications can use Linux file system APIs (e.g. fopen, fwrite, fread). Linux file system will translate file operations to block operations and call device driver through block device ioctls (e.g. block_read, block_write). Device driver converts block operations into EFI defined block operations format, which consists of synchronization status, block address, number of blocks, read/write, and disk size. Three phases on the right-hand side of figure 4 shows this workflow.

In figure 4, EFI Virtual Disk Monitor checks shared memory for block operations. After receiving block operation information, it checks the files access rules, and then writes blocks to or reads blocks from disk image. Every block operation must be checked for access rules to ensure the security of virtual disk.

This program model supports two image loading methods. One is called EFI ram-disk for high speed: disk image can be loaded into EFI memory when disk is mounted in Linux and only write back is allowed when disk is unmounted in Linux. The other is called EFI USB-disk for high security and reliability: data is directly read from and written to a USB drive. A comparison of the two methods is given in the next section.

4.3 Mechanism

EFI Virtual Disk Monitor (VDM) consist of a FAT [8] file system module and a security module. Processing details of VDM is shown in Figure 5. Security module functions on access rules and handles tags on file-to-block mapping table. File-to-block mapping table is maintained by file system module.

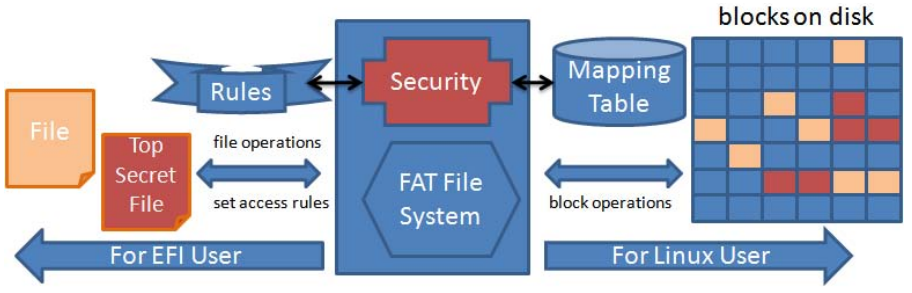


Fig. 5. Mechanism of Virtual Disk Monitor

When a user assigns an access rule to a file, file system module will convert filename to a series of block number and security module tags access rule in the mapping table. Mapping table will be checked on every block write/read request. If any rule on a file is violated, monitor will show a message in EFI and stop associated operations. For EFI users, Virtual Disk Monitor acts as a normal file system with user assigned file access rules. The presented design supports six rules: 1) file system read-only. 2) file read-warning. 3) file write-prohibit. 4) block read-warning. 5) block write-prohibit. 6) display all write/read operations. For Linux users, Virtual Disk Monitor acts as a block device. It only support block write/read operations.

5 Experiment

In this section, two sets of experiment results are presented. Test hardware platform is based on Intel Lakeport develop platform. It consists of Pentium D 3.2GHz, 1GB DDR2 RAM, Quantum Fireball 6.4GB PATA hard disk, and SanDisk Cruzer Micro (512MB) USB disk. Software platform consists of EFI V1.10 with multi-core support, modified Linux kernel 2.6.13 and Iozone 3.283.

Write speed tests were performed for EFI ram-disk, EFI USB-disk and physical hard disk. Implementation methods of EFI ram-disk and EFI USB-disk are described in subsection 4.2. Iozone file system benchmark tool [15] is used for all tests. This benchmark generates and measures a variety of fie operations.

Test procedures for EFI ram-disk and EFI USB-disk:

1. load disk image into virtual disk monitor in EFI
2. insmod EFI virtual disk module in Linux
3. mount virtual disk in Linux
4. mkdosfs with parameter -F16 is executed to format virtual disk in Linux
5. copy and run Iozone test in Linux
6. umount virtual disk in Linux
7. rmmod EFI virtual disk module in Linux
8. unload disk image in EFI

Physical hard disk is tested by copying and running iozone test in a void directory.

5.1 Test Case 1

32MB disk image is used in this test. Write speed tests were performed for EFI ram-disk, EFI USB-disk and physical hard disk. Parameters of Iozone are: -Ra -p -o -s 512k -y 4k -i 0 -i 1 -f test.dat. Parameter -p was used to purge the processor cache before each file operation and -o to force all writes to the file to completely go to disk. Figure 6 shows the results.

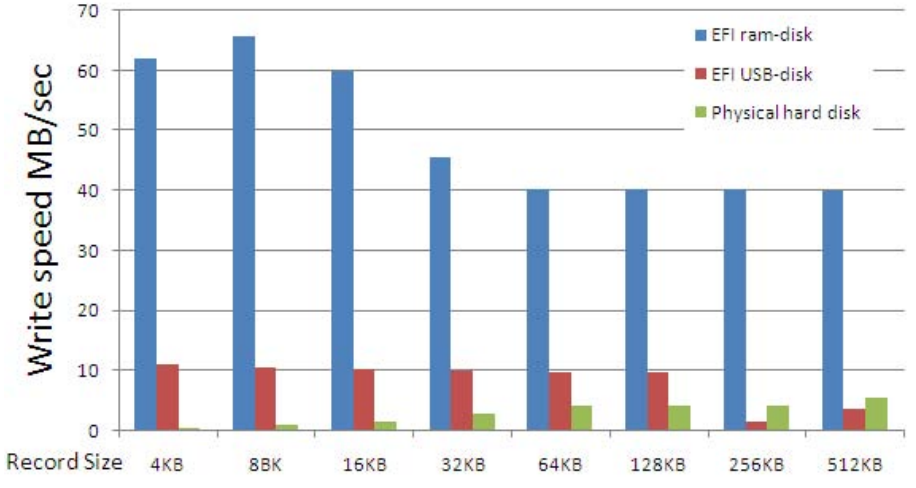


Fig. 6. Iozone Write Performance on 32MB Disk Image

Write speed for EFI ram-disk method is about 48MB/s on average. The speed is unstable when test record size is below 64KB. Write speed for EFI USB-disk method is about 8MB/s on average, but drops fast when test record size is larger than 128KB. Write speed for physical hard disk is steadily up to 5MB/s on 512KB record size and 3MB/s on average. In general, the performance of EFI ram-disk method is about 6 times faster than the performance of EFI USB-disk method on stable speed. The performance of physical hard disk is a little faster than the performance of EFI USB-disk method when block size is larger than 256 KB.

5.2 Test Case 2

256MB disk image is used in this test. Write speed tests for EFI ram-disk and physical hard disk are tested. Parameters of Iozone are: -Ra -s 64m -y 4k -i 0 -f test.dat. Figure 7 shows the results.

Write speed for EFI ram-disk method is about 50MB/s on average and it is stable during the tests. Speed for physical hard disk is about 17MB/s on average and it is also stable. This shows that the average write performance of EFI ram-disk method is about 3 times faster than the performance of physical hard disk method.

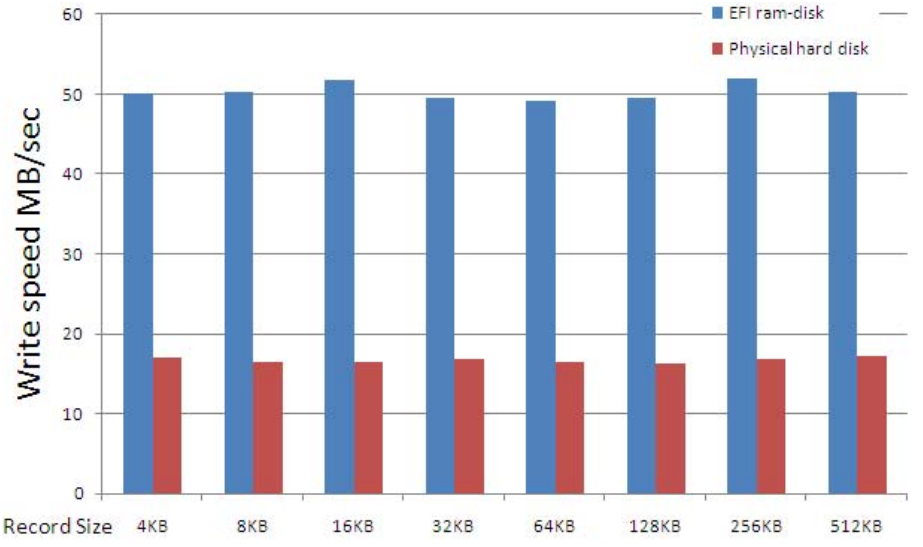


Fig. 7. Iozone Write Performance on 256MB Disk Image

6 Conclusion

In this paper, EFI shows great extensibility in support multi-core architecture and support virtual device for monitored OS. In the future, increasing write speed for EFI USB-disk performance by optimizing EFI USB driver and file library is planned. Furthermore, a plan to design other virtual devices for monitored OS and use more sophisticated security rules on virtual devices will also be considered.

Acknowledgments

We appreciate Intel Corporation for providing hardware platform and EFI environment. Also, a lot of useful supports and suggestions come from Intel EFI/Tiano Team. We would like to thanks anonymous reviewers for their valuable suggestions to improve this manuscript

References

1. Intel. Extensible Firmware Interface Specification Version 1.10 (December 2002), http://developer.intel.com/technology/efi/
2. Intel. Multi-Core Overview, http://www.intel.com/multi-core/overview.htm
3. Intel. Tiano Architecture Specification Version 0.7 (June 2002), http://www.tianocore.org
4. Ruemmler, C., Wilkes, J.: An Introduction to Disk Drive Modeling. IEEE Computer , 17–28 (March 1994)

5. Garfinkel, T., Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection. In: Proceedings of the Internet Society's 2003 Symposium on Network and Distributed System Security (February 2003)
6. Zhang, Y., Gu, Y., Wang, H., Wang, D.: Virtual-Machine-based Intrusion Detection on File-aware Block Level. In: SBACPAD 2006. Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing, pp. 185–192 (2006)
7. Quynh, N.A., Takefuji, Y.: A Real-time Integrity Monitor for Xen Virtual Machine. In: ICNS 2006. Proceedings of the International conference on Networking and Services, p. 90 (July 2006)
8. Microsoft Extensible Firmware Initiative FAT32 File System Specification 1.03
9. Inoue, H., Ikeno, A., Kondo, M., Sakai, J., Edahiro, M.: VIRTUS: A New Processor Virtualization Architecture for Security-Oriented Next-Generation Mobile Terminals. In: Proceedings of the 43rd annual conference on Design automation table of contents, Annual ACM IEEE Design Automation Conference, pp. 484–489 (2006)
10. Hiroaki, I., Naoki, S.: FIDES: A Multi-Core Platform to Enhance Robustness of Embedded Systems. NEC Technical Journal 1(3) (July 2006)
11. Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P., Neugebauer, R.: Xen and the art of virtualization. In: Proceedings of the ACM Symposium on Operating Systems Principles (October 2003)
12. Popek, G.J., Goldberg, R.P.: Formal requirements for virtualizable third generation architectures. Communications of the ACM 17(7), 412–421 (1974)
13. Rosenblum, M., Garfinkel, T.: Virtual Machine Monitors: Current Technology and Future Trends. IEEE Computer, 39–47 (May 2005)
14. Smith, J., Nair, R.: Virtual Machines: Versatile Platforms for Systems and Processes. Elsevier Science & Technology Books (May 2005)
15. Norcott, W.D.: Iozone Filesystem Benchmark User Manual (2003), http://www.iozone.org

An Optimal Design Method for De-synchronous Circuit Based on Control Graph*

Gang Jin, Lei Wang, Zhiying Wang, and Kui Dai

School of Computer Science,National University of Defense Technology, Changsha, 410073, China
jingang@nudt.edu.cn

Abstract. De-synchronous is a very useful method to design asynchronous circuit automatically from synchronous description of circuits. This paper introduces an optimal design method based on Control Graph which is an abstract model of the de-synchronous circuit. The main purpose of this optimal design method is to reduce the extra overhead in the area of the de-synchronous circuit. The optimization algorithm takes the performance evaluation function based on the Control Graph of the de-synchronous circuit as its heuristic function. The performance evaluation function presented in this paper is a linear programming problem. In the end of this paper, the optimal method is applied to a set of benchmark circuits. The number of the local controllers in these circuits is markedly reduced by 54%, and the number of C-elements that is required to construct the handshake circuitry between local controllers is also reduced by 76.3%. So the entire area of the circuit is sharply reduced. Because this design method is directed by the performance evaluation function of the circuit, there is no penalty in performance of the de-synchronous circuit.

Keywords: de-synchronous, asynchronous, performance evaluation, algorithm, control graph, Petri-net.

1 Introduction

Along with the scale of chip is getting larger and larger, the clock skew becomes more and more serious. In order to solve the clock skew problem, a large balance clock tree needs to be constructed, involving much area and consuming much energy of the circuit. Compared with synchronous circuits, the power dissipation of asynchronous ones are really small. As the different parts of the circuit operate at different speed and switching activity, the Electro-Magnetic Compatibility has increased. With their interfaces free from global constraints such as operating frequency, asynchronous circuit provides inherent modularity, which is the major advantage of the asynchronous design methodology. However, there are some disadvantages related to asynchronous circuit. On the one hand, there are no

* Supported by the National Natural Science Foundation of China under Grant No. 90407022.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 70–79, 2007.

good CAD tools for asynchronous circuit design; on the other hand, the design of asynchronous circuit is really complex since there is no global control signal in the circuit.

In order to avoid the disadvantages of the asynchronous design style, the concept of de-synchronous has been brought up [1], whose essential idea is that the design starts from a standard synchronous synthesized circuit, and then the global clock network is directly replaced by a set of local controllers. All steps of this design flow can be implemented within standard CAD tools. This method works efficiently in dealing with the difficulties which will be encountered when adopting a pure asynchronous design style.

When adopting de-synchronous design methodology, a local handshake circuitry should be inserted into the circuit in order to take place the global clock. So an extra overhead, i.e. area, will be introduced to the circuit. The major purpose of this paper is to reduce this area overhead. A abstract model is developed to represent the control path of the circuit, based on which an algorithm is introduced to reduce the area of the de-synchronous circuit.

In Section 2, the related works of this paper are introduced. The concept and design flow of the de-synchronous design style are explained in Section 3. In Section 4 we use an abstract model—Control Graph based on Petri-net to model the de-synchronous circuit. In Section 5, based on Control Graph of de-synchronous circuit, an optimization algorithm is developed to combine the local controllers in order to reduce the entire area of the circuit, which is directed by the performance evaluation function. In Section 6, the algorithm is applied to a number of benchmark circuits. Section 7 is the conclusion part.

2 Related Work

The idea of generating local control signals for a synchronous latch-based circuit is proposed by Sutherland in his Turing award lecture [2]. The micorpipeline theory has been adopted in several designs [3] and CAD tools [4,5,6].

Theseus Logic has developed a design method [7] that uses the commercial EDA tools to synthesize and optimize the datapath, and directly translates the control path into an asynchronous implementation. But this approach suffers from high overhead and requires the non-standard HDL style.

Liner and Harden have introduced a method that replaces each logic gate with an equivalent sequential handshake asynchronous circuit, where the synchronization information is encoded into the code of data using an LEDR delay-insensitive code [8]. This approach also have an expensive overhead.

The similar work as this paper is presented by A. Davare in [9]. He also directly replaces the global clock network with local controllers. But in that paper, he only introduces a simple method to optimize the circuit without concerning the circuit performance.

Our group has also done a lot of works on de-synchronous circuit design. We have presented a de-synchronous circuit design flow [10,11] based on macrocell. This design flow tries to compatible with current EDA tools for synchronous

design, which makes it easy to design asynchronous circuits. Based on this design methodoloy, we have designed a 32-bits asynchronous multiplier in $0.35\mu m$ process. Compared with the synchronous partner, our design of multiplier has smaller area, lower power dissipation and higher performance.

3 Design Step for De-synchronous Circuit

Generally speaking, the design based on flip-flop will need more complex control circuitry, which will lead to an extra area overhead. In this paper, we translate each flip-flop to a pair of master-slave latches, because latch-based design will be smaller and faster. In [1], a design flow of de-synchronous circuit has been introduced. All steps of this flow starting from a flip-flop-based synchronous circuit that can be implemented with standard CAD tools. The de-synchronization method proceeds in the following three steps:

1. Splitting each flip-flop into a master-slave latch pair.
2. Generating matched delay for combinational logic.
   Serving as a completion detector for the corresponding combinational block, each matched delay must be greater than or equal to the delay of the critical path of the corresponding combinational block.
3. Implementing the local controller corresponding to each latch.
   For each latch of the latch-based synchronous circuit, a local controller will be inserted into the circuit for generating the control signal. The local controllers communicate with each other over handshake. Request signals from predecessors are delayed by the matched delay generated in the previous step.

4 The Model of De-synchronous Circuit

In fact, the data path of the de-synchronous circuit presented in this paper has no difference with its synchronous partner, so the major design concern should be paid to the design of the control path of the circuit. Based on the work in [12] , this paper introduces an abstract model of the control path of the de-synchronous circuit—Control Graph. The Control Graph takes a weighted directed graph to represent the local controllers corresponding to the latches in the circuit and the handshake circuitry between the local controllers. The definition of the Control Graph is:

Definition 1. *Control Graph*
A Control Graph is a 4-tuple, $< V, F, W, P >$*;* $< V, F >$ *is a directed graph,* $P : V \mapsto \{even, odd\}$ *is a polarity function,* $W : F \mapsto R$ *is a weighted function.*

In the directed graph $< V, F >$, V is a set of all vertices in this graph, and each vertex in V represents a local controller in the de-synchronous circuit; F is a set of all edges in the graph, and each edge in F represents a connecting relationship of two local controllers, in other words, these two local controllers

should synchronize with each other. In a real circuit, an edge also indicates that there is a combination logic path between the two latches. The polarity function P assigns a polarity to each vertex of $< V, F >$ according to the type of corresponding latch, such as master or slave. The weighted function assigns a real number to each edge of $< V, F >$, which indicates that the worst case delay associates with corresponding combination logic path of this edge.

The construction process of the Control Graph is as follows:

1. Using logical synthesis tools to synthesize the circuit, getting the gate-level net-list of the circuit.
2. Inserting a new vertex to the Control Graph for every latch in the gate level net-list.
3. Determining the connecting relationship between the vertices in the Control Graph, i.e. determining the predecessors and successors of every vertex in the graph. For each vertex v, all vertices that are connected to the input ports of the combination block whose output port is connected to the input ports of v construct the predecessor set of v, $pre(v)$. The successor set $post(v)$ can be determined in same way.
4. Determining the polarity of each vertex in the Control Graph. For vertex v, if the latch corresponding to v is a master latch, define $P(v) = even$, else define $P(v) = odd$.
5. Determining the weight of edge in the Control Graph, i.e. the weighted function W. The worst case delay of each combinational path corresponding to the edge of the Control Graph can be calculated by STA tools. This delay is assigned to the edge as a weight.

An example circuit and its Control Graph are illustrated in Fig 1.

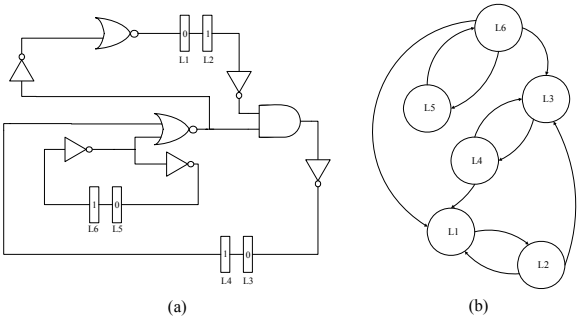


Fig. 1. An example circuit and its Control Graph

The circuitry of control path can be automatically derived from the Control Graph of the de-synchronous circuit. For each local controller v, there is a pair of req_{in}, ack_{in} signals, and a pair of req_{out}, ack_{out} signals, where req_{in}, ack_{in} are the handshake signals with the local controllers in set $pre(v)$; req_{out}, ack_{out} are the handshake signals with the local controllers in set $post(v)$. All req_{out} of the

local controllers in $pre(v)$ are combined by a multi-input C-element to generate the req_{in} of the local controller v; All ack_{out} of the local controller in $pre(v)$ are directly connected to the ack_{in} of the local controller v.

5 Optimization of the Control Path

Based on the Control Graph of the de-synchronous circuit, this paper develops an optimization algorithm to reduce the area of the de-synchronous circuit.

The control path of the de-synchronous circuit is made up by the local controllers and the handshake circuitry between them. The de-synchronous circuit which avoids the area overhead of the global clock introduces the new area overhead of local controllers and the handshake circuitry between them. As the data path of the de-synchronous circuit is the same as its synchronous partner, reducing the area of the control path of the de-synchronous circuit will be important to reduce the entire area of the whole circuit.

We can naturally combine the control signals of a set of latches with a single control signal. One local controller generates the latch signal of a set of latches, by which the number of the local controllers in the control path will be decreased and in turn reducing the whole area of the circuit. According to this concept, we develop an optimal method based on the combining control signals of latches.

5.1 Combining Control Signals of Latches

The meaning of combining control signals of latches is that the control signals of several latches are generated by a single local controller, by which the area of the circuit will be reduced.

In fact, combining control signals of latches is to combine the vertices in the control graph. Because the mapping between the vertex in the control graph and the latch controller in the circuit is one to one, we can use the combination of the vertices in the control graph to represent combining control signals of latches. In this way, we can combine the vertices in the control graph to achieve the purpose of combining the local controllers in the circuit. When two vertices in the control graph are combined, the following rules must be followed:

- Only when two vertices have the same polarity, they can be combined, i.e. $P(u) = P(v)$; After the combination, a new vertex w will be inserted into the control graph.
- $pre(w) = pre(u) \bigcup pre(v)$.
- $post(w) = post(u) \bigcup post(v)$.
- The weight of each edge remains unchanged.

5.2 The Performance Evaluation Function Based on Control Graph

Definition 2. *Average Cycle Time*[13]

The average cycle time of an asynchronous circuit is the longest average cycle time among all circles in the timed Petri-net model corresponding to this circuit.

The average cycle time is a performance evaluation parameter of the asynchronous circuit. De-synchronous circuit is a kind of asynchronous circuit, so this parameter can also be the performance evaluation parameter of the de-synchronous circuit.

Definition 3. *Timed Petri-net*

Timed Petri-net is defined as a 5-tuple $N =< P, T, F, \Delta, M_0 >$, *where* $P = \{p_1, p_2, ..., p_m\}$ *is the non-empty and finite set of place,* $T = \{t_1, t_2, ..., t_n\}$ *is the the non-empty and finite set of transition,* $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relationship,* $\Delta : T \mapsto R$ *is the execution time function of transition,* $M_0 \subseteq P$ *is the initial marking of the Petri-net.*

The timed Petri-net of a de-synchronous circuit can be derived from the Control Graph of the circuit. The procedure is:

1. Every vertex in the Control Graph has been extended to a substructure in timed Petri-net. This substructure is constructed by two transitions and one place; one transition is called input transition which can have several inputs and only one output connecting to the input of the place; the other is called output transition which can have several output and only one input connecting to the output of the place.
2. Each edge in the Control Graph become a place in the timed Petri-net, whose input is connected to the output transition of the extended substructure derived from the source vertex of this edge; the output is connected to the input transition of the extended substructure derived from the target vertex of this edge.
3. The place in the substructure produced in step 1 can present the vertex in Control Graph. Each place corresponding to the odd vertex in the Control Graph should be included in initial marking M_0.
4. For each transition t_i, the corresponding transition execution time θ_i is the maximum delay of the edges which input to t_i.

It is easy to be confirmed that the timed Petri-net derived from above procedure is live and safe.

[14] has pointed that the bottom bound of average cycle is:

$$\tau = \max_i \{ \frac{y_i^T (C^-)^T D x}{y_i^T M_0} \}$$

where $C^- = [c_{ij}^-]_{m \times n}$, c_{ij}^- is the weight of directed arc from transition j to place i; D is the diagonal matrix constructed by θ_{ii}, which is the execution time of the transitions t_i in timed Petri-net; M_0 is an array having the same number of elements as the place set of timed Petri-net, which contains the initial number of tokens kept in the corresponding place.

If the Petri-net is a marked graph (a subclass of Petri-nets[15] that can model decision-free concurrent systems), the maximum average cycle time can be gained

by solving the below Linear Programming problem.

$$\begin{gathered} T = \max Y^T (C^-)^T \theta \\ C \cdot Y = 0 \\ st.\ Y^T \cdot M_0 = 1 \\ Y \geq 0 \end{gathered}$$

This method is very fast, so it works in dealing with large scale problems. For a de-synchronous circuit whose Control Graph is C, we take the average cycle time $T(C)$ calculated in this method as the performance evaluation function of this de-synchronous circuit.

5.3 The Optimization Algorithm

According to the polarity of the vertex in the Control Graph, the latches in the circuit can be divided into two subsets. Only vertices with the same polarity can be combined. In the extreme case, it will result in a circuit with only two local controllers, one for the master latches and the other for the slave latches, which behaviors just like the synchronous partner and eliminates the benefit of the asynchronous one. In fact, the purpose of combination of latches is to find a best partition of these two subsets to achieve the best balance between the area and the benefit of asynchronism.

In order to find the exact optimization result, it is necessary to traverse every partition of the two subsets of the vertices, which is unacceptable for it is a NP-hard problem. Therefore, a polynomial time algorithm to find an approximate optimal solution is introduced in this paper.

To prevent the benefit of asynchronism lost, a threshold is introduced to the optimization procedure. The threshold means the up-bound of the number of latches controlled by a single local controller. The larger value the threshold assigned, the more local controller can be combined, the more benefit of asynchronous is lost, vice versa. The algorithm for combining control signals of latches is as follows:

Algorithm 1 (combining control signals of latches). *Given a Control Graph* $C =< V, F, W, P >$. $V_{deleted}$ *is a set that keeps the vertices deleted during the optimization. Set* $n = |V|$. *It maintains an array* $(\theta_1, \theta_2, ..., \theta_n)$, *where* θ_i *present the times which* v_i *has been combined.* Θ *is the threshold assigned to algorithm. The algorithm is as follows:*

1. *Set* $V_{deleted} = \emptyset$;
2. *Set* $i = 1$, $\tau_{min} = \infty$;
3. *Set* $j = i + 1$;
4. *If* $v_i \in V_{deleted}$, *then jump to* 5. *If* $v_j \in V_{deleted}$ *or* $\theta_i + \theta_j > \Theta$, *then jump to* 4. *Combine* v_i *and* v_j *of* C *to produce a new control graph* C', *if* $T(C') \leqq T(C)$, *then set* $\tau_{min} = T(C')$, $i_{min} = i$, $j_{min} = j$. *If* $j \geqq n$, *then jump to* 5, *otherwise* $j = j + 1$ *and jump to* 4.
5. *If* $i > n$, *then jump to* 6, *otherwise* $i = i + 1$ *and jump to* 3;

6. *If* $\tau_{min} < \infty$, *then combine* $v_{i_{min}}$ *and* $v_{j_{min}}$ *to produce the new control graph* C *and set* $V_{deleted} = V_{deleted} \cup \{v_{j_{min}}\}$.

The core step of this algorithm is step 4, in which $T(C)$ has been calculated. $T(C)$ is a linear programming problem. If we chose the Karmarker's algorithm to solve the linear programming problem, the time complexity is $O(n^{3.5})$ [16], where n is the number of variables in this problem, i.e. the place of timed petri-net corresponding to the circuit. The iterative depth of our optimization algorithm is 2, so the time complexity of the algorithm is $O(n^{5.5})$. The time complexity of this algorithm is polynomial time.

6 Experiment Result

In order to evaluate the effectiveness of the algorithm presented in this paper, several experiments are conducted on some benchmark circuits and describing in the following part.

We choose a subset of the ISCAS'89 benchmark circuit sets, 9 circuits of which are chosen. To determine the effect of the threshold, we choose different threshold values to run the algorithm on these set of benchmark circuits. The reduction of the number of the local controllers is illustrated in Table 1. Since the control path is made up by these local controllers and the handshake circuitry between them, and the major part of the handshake circuitry is C-elements, the total area of the control path is mainly composed by the area of the local controllers and the area of the C-elements. For this reason, we also illustrate the number of C-elements needed by the circuit. In the experiment, we investigate the result when threshold is $2(\Theta = 2)$ and $3(\Theta = 3)$.

Table 1. The experiment result of combining control signals of latches

	Original			Optimized($\Theta = 2$)			Optimized($\Theta = 3$)		
Circuit	Vertex	Edge	C-element	Vertex	Edge	C-element	Vertex	Edge	C-element
s27	6	10	4	4	6	2	4	6	2
s298	28	83	55	16	38	22	12	25	13
s344	30	93	63	16	40	24	12	35	23
s349	30	93	63	16	40	24	12	35	23
s386	12	42	30	6	12	6	4	6	2
s420	32	152	120	14	44	28	12	30	18
s510	12	42	30	6	12	6	4	6	2
s526	42	165	123	22	77	55	14	48	34
s1448	12	42	30	6	12	6	6	12	6

From the experiment result presented above, we can see that the number of vertices and edges are both decreased, and the C-elements required by the circuit are also dramatically reduced. We can also see that when $\Theta = 2$, the number of local controllers in the circuit is totally reduced by 37.9%, the number of

C-elements is totally reduced by 66.6%, and when $\Theta = 3$, the number of local controllers in the circuit is totally reduced by 54%, the number of C-elements is totally reduced by 76.3%.

Along with the reduction of the number of the local controllers in the circuit, the fan-in and fan-out of a single local controller may be increased, which may cause some extra overhead of area introduced into the circuit. In Table 2, we illustrate the average fan-in and fan-out of these circuits before and after optimization.

Table 2. The average fan-in and fan-out of the benchmark circuits before and after optimization

	Original	Optimized	
Circuit	Avg. fan-in/out	Avg. fan-in/out($\Theta = 2$)	Avg. fan-in/out($\Theta = 3$)
s27	2.67	3.00	3.00
s298	3.96	4.15	4.42
s344	4.10	4.38	5.42
s349	4.10	4.38	5.42
s386	4.50	4.00	4.50
s420	5.75	4.75	5.17
s510	4.50	4.00	4.50
s526	4.93	5.41	6.43
s1448	4.50	4.00	4.00

From the result above, we observe that the change of the average fan-in and fan-out is relatively small compared with the notable reduction in the number of the local controllers and the C-elements. In some cases, because of the great reduction of the number of handshakes in the circuit, the average fan-in and fan-out may even be decreased.

7 Conclusions

In this paper, an optimization method to balance the penalties and benefits of de-synchronous circuit is introduced. It is allowed that the control signals of several latches to be combined into a single signal generated by one local controller. In this way, the overhead of the local controllers can be sharply reduced. From the experiment results, we can see that nearly half of the local controllers can be eliminated and nearly 2/3 of the C-elements required to construct the control path can also be eliminated.

The de-synchronous design methodology can improve EMI, and markedly shorten the design cycle of asynchronous circuit. Our optimization method for de-synchronism can conquer the problem that de-synchronism may bring some overhead into circuit. We believe that de-synchronism with our optimization algorithm is a very useful method to design asynchronous circuit before the pure asynchronous design methodology be widely used.

References

1. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: A concurrent model for desynchronization. In: IWLS 2003 (2003)
2. Sutherland, I.E.: Micropipelines. Communications of the ACM 32 (1989)
3. Furber, S.B., Garside, J.D., Gilbert, D.A.: Amulet3: A high-performance self-timed arm microprocessor. In: Proc. International Conf. Computer Design(ICCD) (October 1998)
4. Bardsley, A., Edwards, D.: Coompiling the language Balsa to delay-insensitive hardware (1997)
5. van Berkel, K.: Handshake Circuits: an Asynchronous Architecture for VLSI Programming. Cambridge University Press, Cambridge (2001)
6. Blunno, I., Lavagno, L.: Automated synthesis of micro-pipelines from behavioral verilog hdl. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 84–92. IEEE Computer Society Press, Los Alamitos (2000)
7. Ligthart, M., Fant, K., Smith, R., Taubin, A., Kondratyev, A.: Asynchronous dsign using commercial hdl synthesis tools. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 114–125. IEEE Computer Society Press, Los Alamitos (2000)
8. Linder, D.H., Harden, J.C.: Phased logic: Supporting the synchronous design paradigm with delay-insensitive circuitry. IEEE Transactions on Computers 45, 1031–1044 (1996)
9. Davare, A., Lwin, K., Kondratyev, A., Sangiovanni-Vincentelli, A.: The best of both worlds: The efficient asynchronous implementation of synchronous specifications. In: Design Automation Conference (DAC), ACM/IEEE (June 2004)
10. Yong, L., Lei, W., Rui, G., Kui, D., Zhi-ying, W.: Research and implementation of a 32-bits asynchronous multiplier. Journal of Computer Research and Development 43 (November 2006)
11. Gong, R., Wang, L., Li, Y., Dai, K.: A de-synchronous circuit design flow using hybrid cell library. In: ICSICT 2006. Proc. of 8th International Conference on Solid-State and Integrated-Circuit Technology, Madrid, Spain, pp. 149–158. IEEE Computer Society Press, Los Alamitos (2004)
12. Blunno, I., Cortadella, J., Kondratyev, A., Lavagno, L., Lwin, K., Sotiriou, C.: Handshake protocols for de-synchronization. In: Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, Shanghai, China
13. Wang, L., Zhi-ying, W., Dai, K.: Cycle period analysis and optimization of asynchronous timed circuits. In: Jesshope, C., Egan, C. (eds.) ACSAC 2006. LNCS, vol. 4186, pp. 502–508. Springer, Heidelberg (2006)
14. Wang, L.: Design and Anslysis Techniques of Asynchronous Embedded Microprocessors. Ph.d. thesis, National University of Defence technology, Changsha (September 2006)
15. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 541–580 (April 1989)
16. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: Proceedings of the 16th Annual ACM Symposium on Theory of Computing, pp. 302–311 (April 1984)

Evaluating a Low-Power Dual-Core Architecture

Yijun Liu, Pinghua Chen, Guobo Xie, Guangcong Liu,
and Zhenkun Li

The Faculty of Computer
Guangdong University of Technology
Guangzhou, Guangdong, China, 510006
yjliu@gdut.edu.cn

Abstract. With the rapid development of silicon technology, chip die size and clock frequency increase; it becomes very difficult to further increase the performance of silicon devices only by speeding up their clock. We believe a more practical way to increase their speed is to use the abundant transistor resource to implement several cores and make the cores execute in parallel. In the paper, we propose a processor-coprocessor architecture to increase the speed of the most frequently-used short program segments and reduce their power consumption. As these segments dominate the dynamic execution trace of embedded programs, the overall increment of performance and power-saving is significant. A dataflow coprocessor and a RISC coprocessor are implemented for comparison. The experimental results show the dataflow coprocessor is faster and more power-efficient than the other one, due to the fact that the dataflow coprocessor offers natural properties for fine-grained instruction-level parallel processing and its parallelism is self-scheduling. Except for data dependencies, there is no constrained sequentiality, so a dataflow program allows all forms of instruction parallelism.

1 Introduction

Based on the prediction of ITRS (International Technology Roadmap for Semiconductors), the diameter of a silicon die will be larger than 22 millimeter and one chip can integrate more than 40 billion transistors by 2010 [1]. ITRS also predicts that the clock frequency will be 10 GHz or one clock cycle will be 100 pico-second. However, the frequency of global clock seems to reach its limit, because even under the optimal circumstance, electric current can only travel 30 millimeter at the speed of 300,000 m/s in 100 pico-second. It is very difficult to further increase the performance of silicon devices only by increasing their clock frequency. Using the abundant transistor resource to generate more cores and parallelizing different parts of a silicon circuit is a more practical way to increase its performance. Parallization also reduces the power consumption of a silicon circuit since supply voltage can be reduced. For these reasons, on the PC processor market, manufactories encapsulate several processor cores in a single chip to improve speed and power-efficiency. On the embedded market,

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 80–89, 2007.

application-specific components are integrated with general-purpose microprocessors, like ARM processors, in a SoC to increase performance and reduce power consumption.

Based on the analysis of embedded processing, we proposed that embedding a small coprocessor, which supports a small instruction set, within a general-purpose microprocessor can effectively increase its performance and power-efficiency. The coprocessor can be designed using different architectures. In the paper, we will compare a dataflow [2] [3] coprocessor with a RISC-like coprocessor to evaluate their performance and power-efficiency.

The remainder of the paper is organized as following: Section 2 reviews the architecture of the dataflow coprocessor. Section 3 brings forward a RISC-like coprocessor competitor. Section 4 compares the performance and the power consumption of the two coprocessors. Section 5 concludes the paper.

2 The Dataflow Coprocessor Architecture Review

Based on the analysis of a large number of benchmark programs, we found that phenomenon of locality [4] is very apparent in embedded processing. In other words, embedded processors spend most of their execution time in several small segments of their programs. For example, a JPEG program contains totally 148,040 instructions but an ARM9 processor spends 44% of its time executing only 14 instructions when running the program [5]. Following the same rule, an ARM9 processor spends 76% of its time in executing only 52 instructions of a media encoding/decoding program, totally containing 14,780 instructions [5]. These most frequently-executed program segments (or kernels) greatly influence the speed and power consumption of embedded devices. Our idea is to treat them differently — these program kernels can be executed in a fast and power-efficient coprocessor, which supports only a small number of instructions and contains a small memory. Figure 1 illustrated the idea: the most frequently-used program kernels are stored and executed in the small coprocessor, the other parts of the program is still put in the big system memory and executed by a general-purpose microprocessor. The coprocessor memory (CoMem) is designed to be a part of the main memory to alleviate the overhead of data transport. Since the coprocessor supports only a small number of instructions (about 64) and contains a small memory (several KB), the delay and power consumption of data fetching, instruction decoding, memory accessing and execution is much small than those of a general-purpose processor, which supports a large set of instructions (larger than 100) and contains a big memory (several MB). In our design, the coprocessor is three times the speed of the general-purpose processor and uses only 10% of the power consumption. Therefore, if a kernel of an embedded program dominates 50% of the execution time and it can be put and executed in the coprocessor, the overall speed can be sped up by 50% and 45% of the overall power consumption can be saved.

The dataflow coprocessor using a static dataflow scheme [6] is illustrated in Figure 2, which shows a dataflow diagram for a FIR algorithm. As can be seen

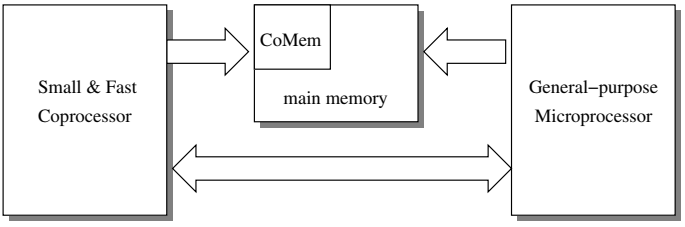


Fig. 1. The proposed microprocessor-coprocessor architecture

from the figure, if a data item is fetched from memory, it will flow through the datapath and the output will be calculated after some time depending on the speed of the multipliers and adders used. The sequence of the operations is not important. As long as the function units (multipliers and adders) have valid inputs, they can process the operations. However, if a stream of data needs to be processed, function units must synchronize with others to prevent overwriting the data values (data tokens) which are being calculated. Thus a dataflow operation should satisfy two requirements:

- Each input port of a function unit should have a valid data token.
- Each output port of a function unit (destination) should be empty, which means the function unit has finished its former calculation (if any) and is ready to process new tokens.

From the example mentioned above, a dataflow graph is constructed from some basic dataflow 'nodes', each representing a basic data processing function. The input arcs of a node represent the inputs for the respective data processing function and the output arcs represent the outputs from the respective data processing function. Normally, a node has two input arcs and one output arc, but the number of input and output arcs can be changed for different purposes. A number of basic nodes are connected together to implement different programs and data flowing through a dataflow graph are referred to as 'tokens'. If two nodes are linked together, the node that issues data tokens is called a 'sender' and that accepting data tokens is called a 'receiver'. Two kinds of dataflow computations are defined depending on who initializes dataflow operations:

- Data-driven computation — operations are executed in an order determined by the availability of input data.

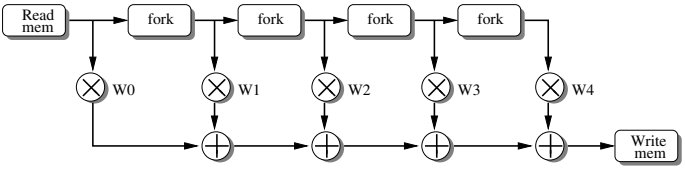


Fig. 2. A dataflow diagram for FIR

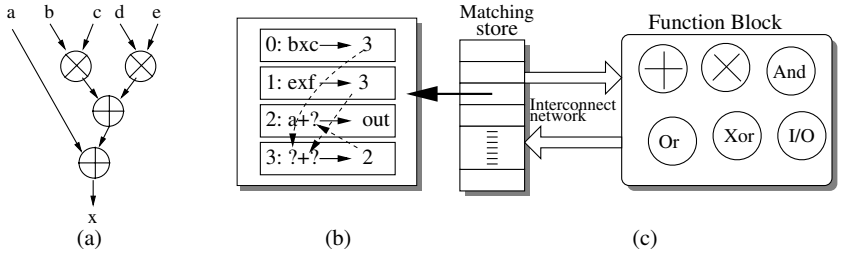


Fig. 3. A general dataflow architecture

– Demand-driven computation — operations are executed in an order determined by the requirements for data.

The coprocessor is designed using an asynchronous scheme [7]. In data-driven computation, senders start dataflow operations, similar to a 'push' channel in asynchronous logic. In data-demand computation, receivers start dataflow operations, similar to a 'pull' channel in asynchronous logic.

Clearly, it is inefficient for a general-purpose data processing device to have a specific datapath for every possible computation such as that shown in Figure 2; a more flexible architecture is needed which can support many different computations. Figure 3(c) shows a basic static dataflow architecture [6] and Figure 3(b) illustrates the mapping of the dataflow diagram in Figure 3(a) onto this architecture. There are three main components in this architecture — a matching store, a data processing unit and an interconnect network. The matching store is used to store instructions and data. It is implemented using a single-port RAM. The data processing unit contains several function blocks where data are processed and an Input/Output module to access main memory or communicate with a CPU. The interconnect network sends data tokens to the function blocks and results back to the specified positions in the matching store.

Figure 4(a) illustrates the structure of the instruction format in the matching store. OPcode indicates the operation of an instruction. Different from those of conventional RISC microprocessor, the dataflow instructions contain two operands (like immediate values in RISC CPUs); a RISC instruction normally indicates the addresses of the registers holding the operands. The Ctr-bits are used to indicate the status of operands and destination to guide data flow. If the two operands of an instruction are both valid and its destination is empty, the instruction becomes 'active'. The interconnect network will 'fire' the active instructions if the corresponding function blocks are free. The 'fired' instruction will send the result to the destination and 'activate' other instructions. Figure 4(a) just shows the format of normal data processing instructions. Other instructions, like load/store and jump/branch instructions, are also implemented in the coprocessor [5]. The proposed dataflow coprocessor architecture is shown in Figure 4(b).

The most interesting property of the dataflow coprocessor is its parallelism. Dataflow execution offers natural properties for fine-grained instruction-level

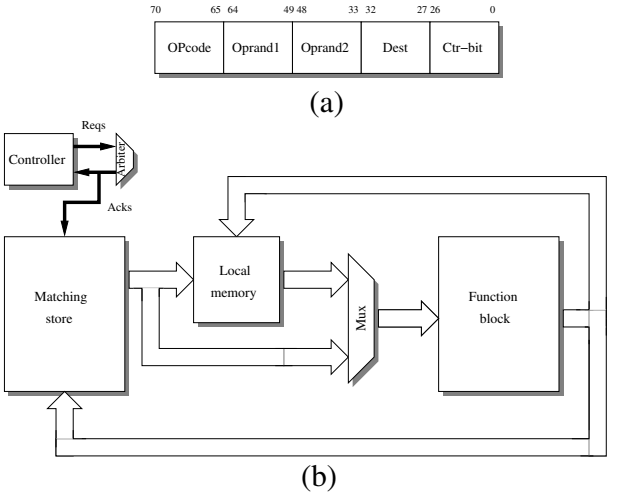


Fig. 4. The proposed dataflow coprocessor

parallel processing. The parallelism of a dataflow program is self-scheduling. Except for data dependencies, there is no constrained sequentiality, so a dataflow program allows all forms of instruction parallelism. Synchronization of different parallel threads is implicit in the form of data interdependencies. With a dataflow scheme, parallelism and synchronization eliminate the need for programmers to use explicit control instructions which manage parallel execution. For example, in Figure 2, fetching a data item from the memory activates two instructions; the two instructions can be sent to the processing unit and execute in parallel as long as enough function blocks exist.

The pure dataflow machines also allow instruction-level parallization, however the big matching stores make it very difficult to implement them efficiently. Our coprocessor only stores and executes the most frequently-used program kernels, thus containing a small matching store (at most hold 64 instructions) as mentioned before. Therefore, it is designed fast and efficiently using an asynchronous logic scheme [5]. The coprocessor is embedded in a general-purpose microprocessor as a 'computation engine' to improve the speed and power efficiency of the most repeatedly executed instruction segments.

3 RISC Coprocessor

To evaluate the performance and power-efficiency of the dataflow coprocessor, a RISC-like counterpart is implemented. The coprocessor instruction set is designed using a straight forward way. The prototype instruction set of the coprocessor includes only three kinds of instructions: data processing, branch and load/store instructions. All the instructions have an instruction-length of 20 bits. The first 5 bits represent what kind operation the processor should execute. The

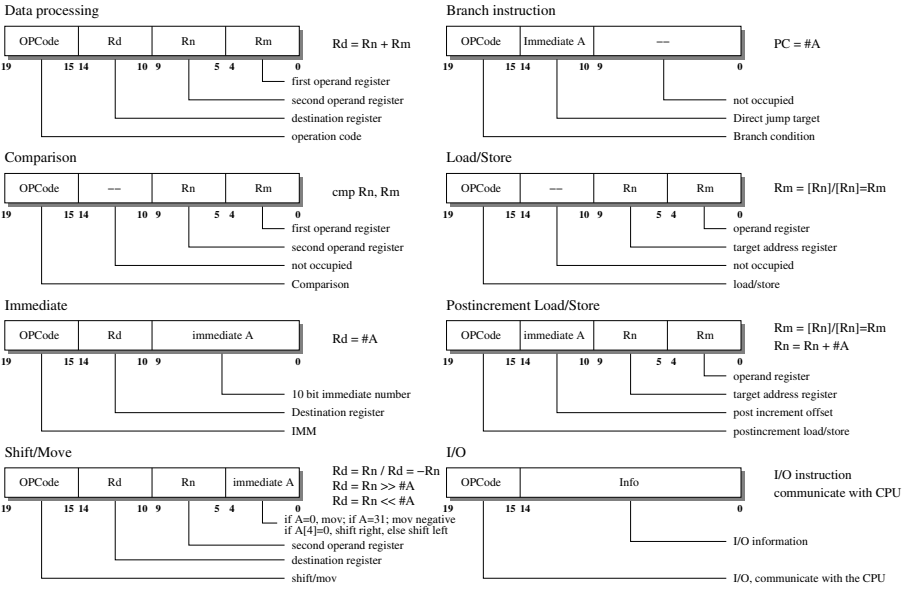


Fig. 5. The structure of the instruction set

unused opcodes can be left for specific data processing applications. Instructions for division, Hamming distance, Counting Leading Zeros (CLZ) and S-box for DES algorithm are possible candidates [8]. The last 15 bits of an instruction are divided into 3 segments with different definitions for different instructions. The structure of the instruction set is shown in Figure 5 and the prototype instruction set of the coprocessor is shown in Table 1. As can be seem from Figure 5 and Table 1, the coprocessor is very small but capable of executing many data processing functions.

Figure 6 (a) and (b) show the proposed RISC architecture of the coprocessor and its pipeline stages. Since the instructions of the coprocessor are very simple, there is no need to include a dedicated instruction decoding stage. Therefore, the pipeline architecture includes only 4 pipeline stages: instruction fetch (IF), register file reading (REG), processing/memory (EXE) and result writing back (WB). Since the load and store instructions in the coprocessor do not need to calculate an operand memory address, the memory fetch/store circuits are put in parallel with the function units which process data. Both the load and store instructions can complete in one cycle. The postincrement load and store instructions can also complete in one cycle by allowing a memory load/store operation to execute in parallel with an address calculation. The postincrement load may cause a slight longer latency than other instructions because two results need to be written back to the register file (one data item from the memory and the post incremented address), but the latency can be tolerated by an asynchronous pipeline.

Table 1. The coprocessor instruction set

No.	Mnemonic	Meaning	No.	Mnemonic	Meaning
0	AND	logic AND	16	BAL	branch always
1	OR	logic OR	17	BEQ	branch if equal
2	XOR	logic XOR	18	BNE	branch if not equal
3	ADD	addition	19	BGT	greater than
4	SUB	subtract	20	BLT	less than
5	CMP	comparison	21	BGE	greater than or equal
6	SM	shift/move	22	BLE	less than or equal
7	MUL	multiplication	23		
8	IO	CPU communication	24		
9	IMM	move immediate	25		
10	LDR	load	26		
11	PLDR	postincrement load	27		
12	STR	store	28		
13	PSTR	postincrement store	29		
14			30		
15			31		

Unlike the classic 5-stage pipeline architecture, which puts the memory load and store operations after the ALU, the coprocessor puts it in parallel with the function units. Since the coprocessor does not allow load/store instructions to have address calculations, putting the memory load/store circuit in an early pipeline stage helps to improve the speed and reduce the energy consumption used by propagating opcodes and addresses through more stages in the datapath.

4 The Experimental Comparison of the Two Coprocessors

The two coprocessors were implemented at a schematic level using a SGS-Thomson 0.18 micron CMOS technology. The datapaths of the coprocessors are both 16-bit. Five benchmarks are used to compare the speed and the power efficiency of the two coprocessor. These benchmarks are:

- $SUM = \sum_{i=1}^{100} i$;
- A 5-tap finite impulse response (FIR) filter with 500 inputs;
- A 5-tap infinite impulse response (IIR) filter with 500 inputs;
- The IDEA encryption algorithm with 500 inputs;
- A 4-point fast Fourier transform (FFT) algorithm with 500 inputs.

Table 2 shows the speed and power consumption of the coprocessors when running these five benchmark programs.

As can be seem from the table, the dataflow coprocessor is 1.6 times faster than the RISC coprocessor, thanks to its natural self-scheduling parallelism. In the dataflow coprocessor, one 'fired' instruction may send results to several destinations and 'activate' several instructions. Therefore the arbiter can find

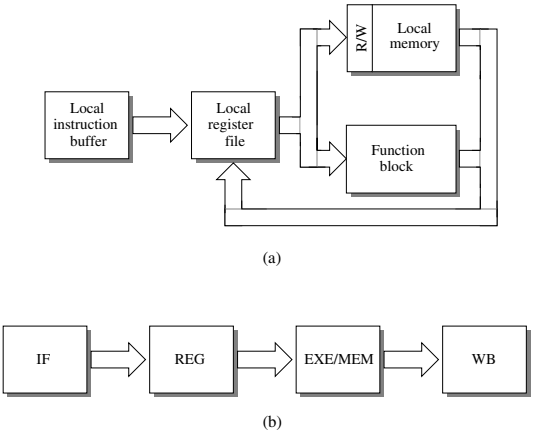


Fig. 6. The organization and pipeline architecture of the coprocessor

Table 2. The comparison of the two coprocessors

Benchmark	Dataflow time	RISC time	Speed Ratio	Dataflow energy/Inst	RISC energy/Inst	energy Ratio
SUM	0.42	0.64	1.6	24.7	48.5	0.51
FIR	7.50	11.28	1.5	35.9	85.5	0.42
IIR	8.52	11.90	1.4	34.6	72.1	0.48
IDEA	14.5	23.20	1.6	29.3	54.3	0.54
FFT	11.7	18.74	1.6	32.8	52.9	0.62
Average	–	–	1.6	–	–	0.51

'active' instructions to fill the pipeline and keep it busy most of the time. On the other hand, the pipeline of the RISC coprocessor stalls from time to time due to pipeline hazards, such as data dependence and branch shadow.

Figure 7 shows the executing segments of the dataflow coprocessor and RISC coprocessor when they execute the FIR program. As can be seen from the figure, the executing segment of RISC coprocessor is highly regular because of its sequential control. For dataflow coprocessor, however, the executing segment is irregular because the arbiter randomly selects active instructions and the executing sequence of instructions is ambiguous. The experiment shows that the dataflow coprocessor is 1.6 times faster than the RISC coprocessor, which also can be seen from the figure.

To sum up, the dataflow coprocessor is faster than the RISC coprocessor because dataflow one can alleviate pipeline stalls due to the data/control dependence of one instruction by selecting other active instructions. How much faster the dataflow coprocessor is than the RISC coprocessor depends on the actual programs. If we add another set of data processing unit in the dataflow coprocessor, its speed will be further sped up.

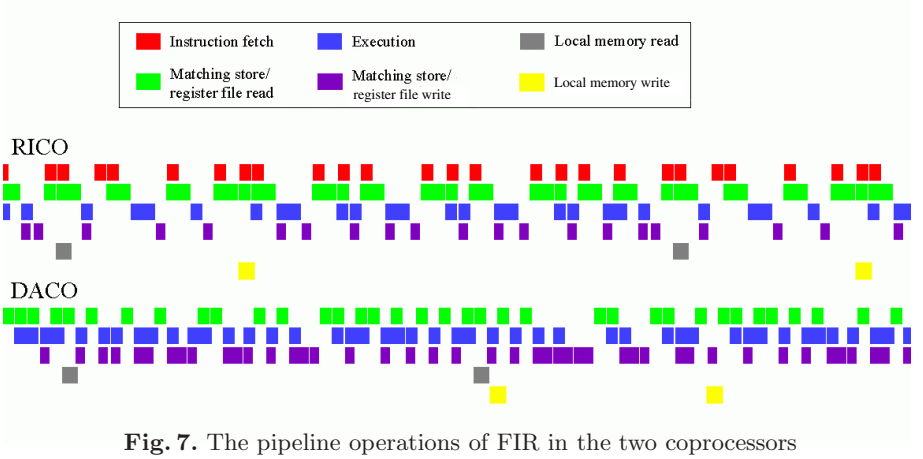


Fig. 7. The pipeline operations of FIR in the two coprocessors

The dataflow coprocessor is more power-efficient than the RISC coprocessor. We found the reason is mainly due to the way of their memory access. The dataflow coprocessor fetches operands (immediate values) together with instructions, thus a one-port RAM can be used to implement the matching store. However, to execute an instruction, the RISC coprocessor must fetch the instruction and read two operands from a three-port register bank, needing three read operations. Three-port register bank uses more energy than one-port RAM and accessing memory three times consumes more power than accessing memory only once. Therefore, the dataflow coprocessor is more power-efficient than the RISC coprocessor.

We compared the speed of the dataflow coprocessor with an ARM9 soft core, it is three time faster than the ARM microprocessor and use only 10% of the power. The main reason is due to the small size of the coprocessor. Based on Amdahl's law [9], if a kernel of an embedded program dominates 50% of the execution time and it can be put and executed in the coprocessor, the overall speed can be sped up by 50% and 45% of the overall power consumption can be saved.

5 Conclusion

In the paper, a dataflow coprocessor is proposed, which is used to be embedded in a general-purpose microprocessor to increase its performance and minimize its power consumption. To evaluate the speed and power efficiency of the dataflow coprocessor, an RISC coprocessor is implemented. Based on experimental results, the dataflow coprocessor is 1.6 times faster than the RISC one and uses only 51% of the power. The speedup is due to the highly instruction-level parallelization of the dataflow scheme. The dataflow architecture is not new and may not fast due to the complex control of big-size matching store. However, we can design a small dataflow coprocessor very efficiently and the dataflow coprocessor can help general-purpose microprocessor to increase speed and reduce power consumption by executing the most frequently-used small program kernels.

References

1. ITRS, International Technology Roadmap for Semiconductors Report, 2006 Update (2006), http://www.itrs.net/
2. Veen, A.H.: Dataflow machine architecture. ACM Computing Surveys (December 1986)
3. Gurd, J., Watson, I.: A data driven system for high speed parallel computer. Computer Design (June 1980)
4. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative approach. Morgan Kaufmann, San Francisco (2003)
5. Liu, Y., et al.: The design of a dataflow coprocessor for low power embedded hierarchical processing. In: Vounckx, J., Azemard, N., Maurine, P. (eds.) PATMOS 2006. LNCS, vol. 4148, Springer, Heidelberg (2006)
6. Dennis, J.B., Misunas, D.P.: Preliminary architecture for a basic data-flow processor. In: Proceedings of the 2nd Annual Symposium on Computer Architecture (December 1974)
7. Sparsø, J., Furber, S. (eds.): Principles of Asynchronous Circuit Design: A systems Perspective. Kluwer Academic Publishers, Dordrecht (2001)
8. Proakis, J.G., Manolakis, D.: Digital Signal Processing: Principles, Algorithms and Applications, 3rd edn. Prentice-Hall Engineering/ Science/Mathematics (1995)
9. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computer capability. In: Proceedings of AFIPS Spring Joint Computer Conference (1967)

Reducing Storage Requirements in Accelerating Algorithm of Global BioSequence Alignment on FPGA

Fei Xia and Yong Dou

Department of Computer Science, National University of Defence Technology, Changsha, P. R. China, 410073
{xcyphoenix,yong_dou}@hotmail.com

Abstract. In the paper, we present storage optimization scheme for hardware accelerating Needleman-Wunsch algorithm. The scheme exploits the characteristics of back-tracking phase in which the back-trace path only travels in a constrained area. Our analysis shows that in addition to logic element resource and memory capacity, the number of RAM blocks is also one of the constrained factors for hardware accelerating bio-sequence alignment. The optimized algorithm only store part of the score matrix to reduce storage usages of FPGA RAM blocks, and implement more processing element in FPGA. We fit our design on FPGA chips EP2S130 and XC2VP70. The experimental results show that the peak performance can reach 77.7 GCUPS (Giga cell updates per second) and 46.82 GCUPS respectively. Our implementation is superior to related works in clock frequency, the maximal PE number and peak performance, respectively.

Keywords: Bioinformatics, FPGA, Global BioSequence Alignment, Needleman-Wunsch algorithm, Hardware Accelerator.

1 Introduction

With the technology development of the genome sequencing, the scale of Gene database expands steeply. In August 2005, the INSDC announced the DNA sequence database exceeded 100 gigabytes and the number of sequence reached over 52 million[1]. It is inefficient to scan the Gene-Bank using traditional software approach. In recent years, FPGA have emerged as performance accelerators capable of implementing fine-grained, potential massively parallelized algorithm for computation-intensive applications. The reconfigurable FPGA chips also enable algorithms to be implemented with different computing structures on the same hardware platform[2]. As a result, hardware accelerating bio-sequence matching attracts much more attention.

After Needleman-Wunsch algorithm was published in 1970, it soon became the standard technique in biological matching, which uses DP-based method (dynamic-programming) and is suitable for global alignment of pair-wise sequence with a certain similarity. It also spawned many variations, including the famous Smith-Waterman algorithm for local alignment.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 90–99, 2007.

Because sequence comparison algorithms based on DP have been proven to produce an optimal alignment, a number of hardware parallel architecture based on the Needleman-Wunsch or the Smith-Waterman algorithm have been proposed for sequence analysis[3],[4],[5],[6],[7],[8],[10] and most of them only concentrate on the scoring phase. Some implementations[5],[11] address on the structure and scale of PE array, but did not consider the storage problem of DP matrix. Works in[3],[6],[7] and[9] discussed how to accelerate the scoring process and enhance the performance scaling, but did not implement the backtracking process. Researches in[4],[8] mapped the backtracking process in FPGA, but did not take the storage optimization of scoring matrix into account.

Since the storage complexity of DP-based method is $O(M \times N)$ for two strings with size M and N. Furthermore, the required port number is linear scale with the size of PE array. With the growth of sequence length, it is difficult to implement both of the scoring and backtracking process of DP matrix in FPGA. To reduce the storage requirement in Needleman-Wunsch algorithm, we present a storage optimization scheme for global bio-sequence alignment applications with backtracking. Based on the analysis to the characteristic of Needleman-Wunsch algorithm, we find that given a fixed scoring method, the backtracking paths always fall into a limited area in DP matrix even in the worst cases, which means storing all elements of DP matrix is unnecessary. Our scheme uses two extra registers for checking address limits, but saves about 50% of storage space in general cases. The saved memory blocks can be used to implement more processing elements. Our experimental results show over 800 PEs can be fitted in an FPGA chip of Altera EP2S130, the maximal speedup reach 5.6 compared to closely related works, and achieve peak performance 77.7 GCUPS.

2 Needleman-Wunsch Algorithm Overview

The basic idea of Needleman-Wunsch algorithm is to use the best alignment of shorter subsequence to build the best alignment of two sequences gradually and recursively. In practice, a matrix F is used to store alignment scores of subsequence. When F is figured out, the scoring process is finished. Needleman-Wunsch algorithm is composed of two phases: firstly, calculate the DP matrix and store the computing trace; secondly, execute trace-back operation according to DP matrix. In this paper we use the convention that the query sequence S with length M is along the vertical dimension and the sequence L with length N in database along the horizontal dimension.

Scoring Phase: Suppose $F(i,j)$ represents the best alignment score between subsequence $S_{1...i}$ and subsequence $L_{1...j}$. The score $F(i,j)$ for grid cell (i,j) is computed following below equations($1 \leq \mathrm{i} \leq M, 1 \leq j \leq N$).

Initialization:

$$\begin{cases} F(0,0) = 0 \\ F(i,0) = F(i-1,0) + P(S_i, -) \\ F(0,j) = F(0,j-1) + P(-, L_j) \end{cases} \tag{1}$$

Recurrence relation:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + P(S_i, L_j) \\ F(i-1,j) + P(S_i, -) \\ F(i,j-1) + P(-, L_j) \end{cases} \tag{2}$$

Thus, we can translate the above recurrence relation (2) into a systolic parallel algorithm executing on linear processing elements, PE, as shown in Fig.1(A).

Each PE calculates a row of DP matrix and stores corresponding trace flag in its local memory. Multiple PEs compute different elements in a line perpendicular to the main diagonal concurrently. In the scoring stage, PE[n]($n \geq 1$) receives score and current character in sequence L from PE[*n-1*] (PE[0]'s input is supplied by array control module). PE[n] calculates the score of current grid, generates trace-back flag and transmits the scoring result and current character to PE[*n+1*]. Finally, in step S5, PE[n] stores the flags into *PE_LM[n]* (PE[n]'s local memory).

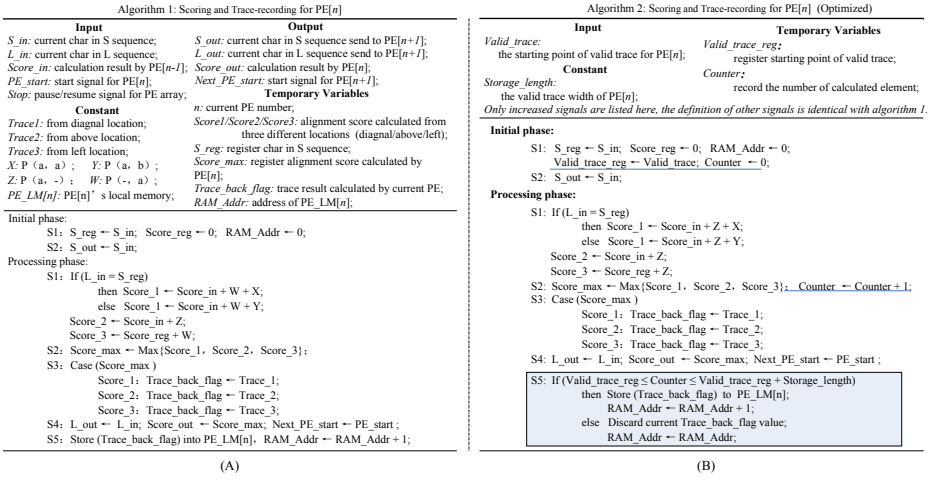


Fig. 1. (A)Scoring and trace-recording algorithm; (B)Optimized scoring and trace-recording algorithm for each PE

Trace-back Phase: After scoring phase, begin backtracking process as shown in Fig.2(A). The start point is set to the low-right element of DP matrix recorded in scoring phase. The trace-back processing can find out the location of the next trace-back point through the current flag. Then set the next point as the current trace-back point until reaches the top-left element of DP matrix.

At the end of backtracking, the path composed by trace-back points is the best alignment. In traditional Needleman-Wunsch algorithm, each PE is responsible for calculating and storing the all elements of corresponding row of DP matrix. The storage requirement is $M \times N$ (M and N are the length of input sequences). With the growth of sequence size, the storage requirements may exceed the

capacity of on-chip memory. In order to implement larger scale scoring and trace-back process on FPGA chips, we present a storage reduction strategy for Needleman-Wunsch algorithm.

3 Storage Optimization Strategy

Given two input sequences S and L, $|S| = M$, $|L| = N$. With universality, we suppose $N > M$. Moreover we using linear gap penalty model and the scoring scheme is shown as follows: $\mathrm{P}(\mathrm{a},\mathrm{a}) = \mathrm{x}, \mathrm{P}(\mathrm{a},\mathrm{b}) = -\mathrm{y}, \mathrm{P}(-,\mathrm{a}) = \mathrm{P}(\mathrm{a},-) = -\mathrm{z}$ $(\mathrm{x},\mathrm{y},\mathrm{z} > 0)$. Parameter G represents the number of gaps in sequence L. R is the number of replace operation in sequence S. The optimized algorithm is shown in Fig.1(B).

The basic calculating process of optimized algorithm is consistent with traditional Needleman-Wunsch algorithm. The main difference lies in S5, where two registers, *valid_trace_reg* and *counter* are used to check the address range so that only valid traces are stored in PE local memory. The former register is filled with starting location of valid trace range in the phase of initialization and the latter records the location of current point of DP matrix. Therefore, Each PE only records partial elements of corresponding row of DP matrix. As a result, the largest memory cost of PE is $N - M + 2G + 1$, the total memory cost of the optimized algorithm is:

$$(N - M + G + 1) \times M, \qquad G \leq \frac{(M - R) \times (x + y)}{2 \cdot z + x} \tag{3}$$

The correctness of formula(3) is proved as follows: (only consider the situation $N > M$; when $N \leq M$, the conclusion is the same). The selection of trace-back path is closely related with the location of gaps inserted in the alignment result. As for trace-back point $F(i,j)$, there are three possible choices in trace selection: vertical path (trace a in Fig.2(A), pointing to element $F(i-1,j)$, it means inserting a gap at the location of L_j in horizontal sequence); horizontal path (trace b pointing to $F(i,j-1)$, it means inserting a gap at the location of S_i in vertical sequence) and diagonal path (trace c pointing to $F(i-1,j-1)$, it means the two sequences generating a match or mismatch at the location of $F(i,j)$).

When $G = 0$, there is no gap in sequence L, that means trace-back path contains no vertical trace, then the gaps in sequence S is $N - M$. So there are only $N - M$ horizontal traces and M diagonal traces in the trace-back path. Whatever the alignment score is, all possible paths must be fall into the shadow parallelogram area in Fig.2(A). The trace 1, 2 and 3 are three possible paths.

Therefore, we can get all the information about backtracking phase recording the elements in the above parallelogram shadow area. Thus the length of PE local memory is $N - M + 1$, and the storage cost of algorithm is $(N - M + 1) \times M$, the proportion of saved storage cost is:

$$T = \frac{M - 1}{N} \tag{4}$$

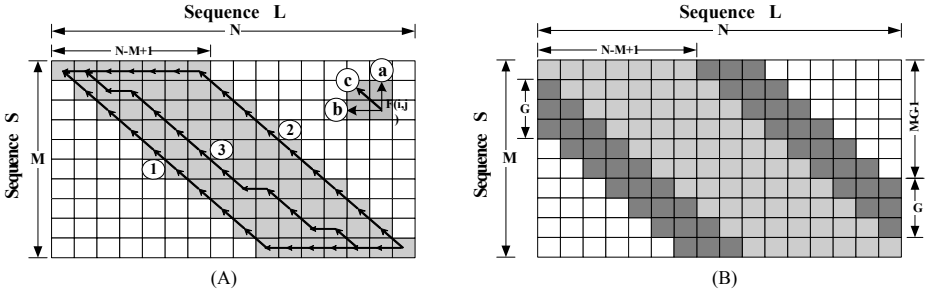


Fig. 2. (A)Valid trace area in DP matrix (G=0); (B)Valid trace area in DP Matrix (G≠0)

When G≠0, there are G vertical traces in backtracking path. Then the path will transcend the shadow parallelogram area in Fig.2(A). Thus it's necessary to extend the recording area. Two gaps matching each other are impossible, so the number of blanks in sequence S is $N - M + G$. There are only four situations in sequence alignment: match, replace, delete and insert (in-del). According to the linear gap penalty model and the scoring scheme above, the matching score is $(M - G - R)\cdot x$, in-del (gap) penalty is $(N - M + G)\cdot z + G\cdot z$ and replace penalty is $R\cdot y$, so the alignment score of pair-wise sequence S and L is $(M - G - R)\cdot x - R\cdot y - (N - M + 2G)\cdot z$. The score in the worst condition is $(N - M)\cdot(-z) - M\cdot y$, (any pair of characters in sequence S and L can't match in this situation). The alignment score in common condition should be greater than the worst case, thus

$$(M - G - R)\cdot x - R\cdot y - (N - M + 2G)\cdot z \geq (N - M)\cdot(-z) - M\cdot y \quad (5)$$

As a result of (5), G has an upper limit:

$$G \leq \frac{(M - R)\times(x + y)}{2\cdot z + x} \quad (6)$$

Therefore, we can obtain all trace-back information by recording only the elements in the shadow area as depicted in Fig.2(B). Thus the largest length of PE local memory is $(N - M + 2G + 1)$, and the whole storage cost is $(N \times M) - (M - G)\times(M - G - 1)$.

According to the analysis above, if the scoring rule is fixed, the gaps inserted in sequence L are limited by the range of G given in formula (6). The proportion of saved storage of optimized algorithm is:

$$T = \frac{(M - G)\times(M - G - 1)}{M \times N} \quad (7)$$

In implementation we should consider not only the capacity of memory and LE (Logic Cell) but also the size of RAM block and the port constrains of FPGA.

Suppose: L_F is number of logic cells in FPGA, C_F is RAM capacity, P_F is the port number of RAM blocks in FPGA, M_C is the capacity of single RAM block,

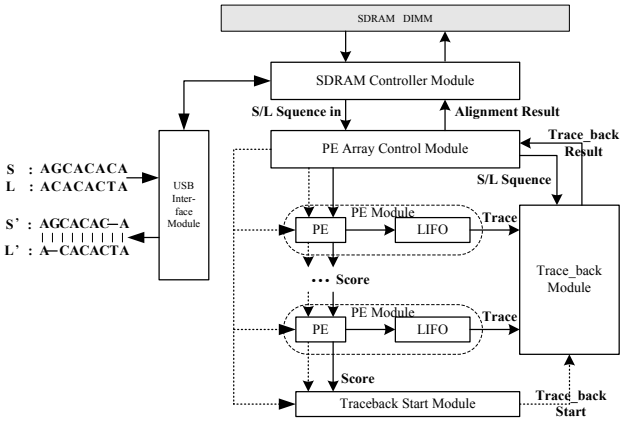


Fig. 3. The structure of N-W algorithm accelerator

P_{PE} is the port number PE uses and L_{PE} represents PE logic cost. The number of PE, N_{PE}, can be fit in a single chip must fulfill the following constrains:

(1) RAM capacity constrain: $N_{PE} \times (N - M + 2G + 1) \leq C_F$;
(2) Logic capacity constrain: $N_{PE} \times L_{PE} \leq L_F$;
(3) Memory port constrain: $N_{PE} \times P_{PE} \leq P_F$;

$$P_{PE} = \left[(N - M + 2G + 1) \cdot {}^{d}/_{M_C} \right] + 1 \tag{8}$$

Where d is storage cost of each element in DP matrix, $(N - M + 2G + 1) \cdot d$ is the memory cost of PE, square brackets means getting the floor of number.

The above analysis is suitable for other scoring rules. The optimized method not only can reduce memory cost without increasing design complexity but also increase the number of PEs. Furthermore, the experimental result shows that the port number of RAM blocks in FPGA is usually the main constraining factor.

4 Design and Implementation

We have implemented the optimized algorithm in the Altera StratixII EP2S130C5 FPGA. The test-bed of our algorithm accelerator includes a FPGA chip, two SDRAM modules and a USB Peripheral Controller. The algorithm core includes PE Array Control Module, PE Array, Trace-back Start Module and Trace-back Module. The structure of N-W algorithm accelerator is shown in Fig.3.

PE module contains score calculation unit (compute element in DP matrix), trace generation unit (generate trace mark) and trace storage LIFO (Last in First out queue, implemented by block RAM) shown in Fig.4(A). The structure of score calculation unit shown in Fig.4(B) consists of three adders and three comparators in terms of formula (2). Since the trace-back marks depend on scoring results, the score calculation becomes the critical path.

Trace-back module is in charge of finding out the valid trace-back flag and generating final alignment. It accesses PE local memory in trace-back phase.

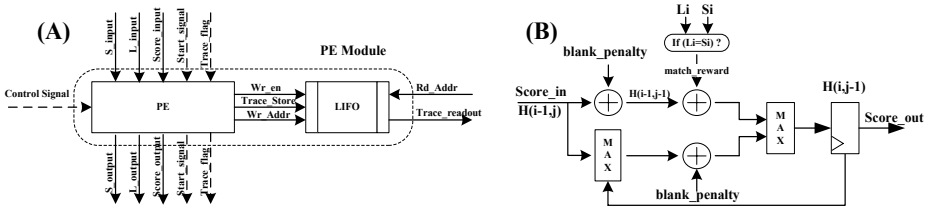


Fig. 4. (A) PE Module Structure and (B) Score Calculation Component

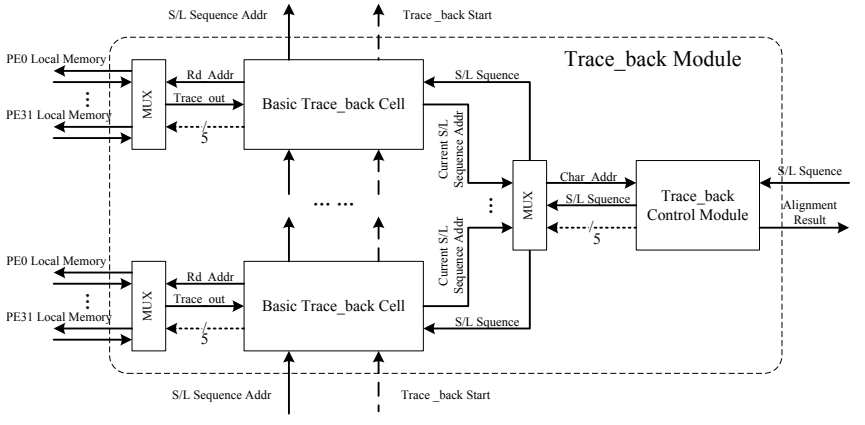


Fig. 5. The structure of trace-back module

When the scale of PE array is large enough, the huge multiplexer becomes the bottleneck in FPGA implementation. To solve the problem, we adopt the well-phased trace-back strategy. The structure of trace-back module shown in Fig.5 is composed of multiple basic trace-back cells and a control module. We divide the linear PE array into several groups so that each basic trace-back cell accesses the corresponding local memory group of PEs and controls the trace-back procedure of current stage (The experiments show that the 32 PEs per group is an optimal choice). The kernel of Basic Trace-back Cell is address generation component, which calculates the address of next trace-back point.

5 Experiments and Performance Comparison

We made experiments with different parameter G. To simplify experiment, we use the following scoring rules: $P(a,a) = 1$, $P(a,b) = -1$ and $P(-,a) = P(a,-) = -1$.

5.1 Reducing Local Memory Requirements

Our optimized storage scheme requires less memory size than traditional algorithm for storing DP matrix. This will save storage space for more PEs

implementation. Fig.6 indicates the experimental result. Given the PE number equals the length of sequence S, M=512 and G, the gaps in sequence L, equals 0, $M/8$, $M/4$ to $3M/8$ respectively. The saving storage percentage can be calculated as equation (7).

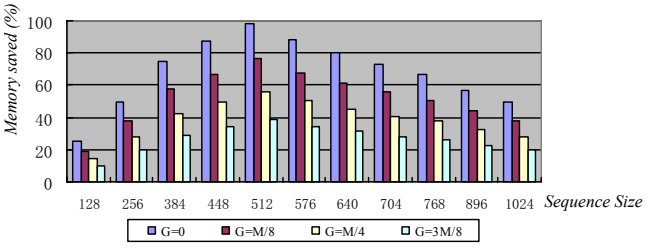


Fig. 6. Proportion of memory saved in different sequence size and parameter G

From the above figure, supposing the lengths of both sequence S and L equal to 512 and no gaps are inserted, our optimized scheme achieves the maximal storage reduction, nearly 99%, compared with traditional algorithm. Even in the worst cases, where the length difference between S and L rises and the inserted gaps also increases to $3M/8$, the reduction percentage still reach about 10%.

5.2 Increasing PE Number

Besides of the limitation in logical resource and memory capacity, the port number of FPGA RAM blocks also constrains the size of PE array. Since each PE occupies one LIFO, which is composed of at least one RAM block of FPGA. The saved storage will provide extra RAM blocks for more PE implementation. Table 1 shows the comparison of PE number implemented in FPGA EP2S130C5 with different length of sequence L.

Table 1. PE number for different sequence size

Sequence L(N)	128	256	512	1024	2048	4096
PEs(Traditional)	928	928	928	780	680	340
PEs (Optimized)	928	928	928	928	780	680

The maximum PE number fitted in FPGA is closely-related to the sequence size. In the condition of $N \leq 512$, the PEs can be fitted in FPGA is limited not by memory but logic resource. With the same FPGA logic resource, the maximal PE number is the same as 928.

But when $N > 512$, the storage factor takes more effects on the scale of PE array. The saved storage can implement more PEs in our optimized scheme than traditional algorithms. The difference between the maximal PE number increases greatly. When sequence length reaches 4096, our scheme can achieves double PE number, as shown in the last column of Table 1.

5.3 Experimental Result

We implemented our optimized algorithm on FPGA StratixII EP2S130F1020C5, supposing $N = 1024$, $M = 800$, $G = M/8 = 100$. The PE local memory capacity is $512 \times 2bit$ occupying two M512 RAM blocks or one M4K block. Fitting 800 PEs consumes 87% logical elements and the clock frequency reaches 97.13MHz, as shown in the first column of Table 2.

Table 2. Performance results and comparison ([*]: the usage of RAM Blocks)

	Ours		ASM[4]	HCP[3]	SRC[11]	PC[4]
FPGA	EP2S130C5	XC2VP70-5	XC2VP70-5	XC2V6000	XC2V6000	XeonPC
PEs Fitted	800	384	303	252	4 Engines	—
LEs (LUT)	87%	73%	—	—	/Chip	
M512 (%)[*]	394 (56%)	BRAMs				N-W
M4K (%)[*]	609(100%)	323 (98%)	—	—	4 Chips	Algo-
Mem Capacity	13%	11%				rithm
Clock (MHz)	97.13	121.93	77.5	55	100	3000
Speed (GCUPS)	77.7	46.82	23.48	13.9	42.7	0.046

Since traditional algorithm needs $N \times M = 800 \times 1024$ memory cells, it is impossible to generate 800 RAM blocks with the capacity of $1024 \times 2bit$ in EP2S130. The optimized approach reduces local memory usage of each PE and saves 50% storage cost, which makes the implementation can be fitted in EP2S130. From Table 2 we also find that the memory usage in our work is only 13%. The reason is that large part of memory capacity in FPGA is implemented by M-RAM block with size of 1Mbits, which can only be used by at most two processing elements. Thus, the bottleneck lies in the number of RAM blocks, not memory capacity for sequence alignment application.

For comparison to related work, we also implement 384 PEs on FPGA chip of Xilinx XC2VP70-5. The result shows our design is superior to the implementation[4] in both PE number and clock frequency. The performance speedup is nearly 2.0. We also compared our performance to the closely related proposals, HCP[3] and SRC[11]. Our implementation achieves the peak performance of 77.7 GCUPS on EP2S130 and the speedup can reach 5.6 and 1.8 relatively.

In addition, we tested the execute time of global pair-wise sequence alignment with backtracking in our FPGA testbed. With sequence length 512, the scanning time of total 1000 sequences is 90.8ms. For the same application on a PentiumIV 2.6 GHz PC, the run time is 31770ms. Hence, our FPGA implementation achieves a speedup of approximately 350.

6 Conclusion

This paper presented the design and implementation of storage reduction strategy of global bio-sequence alignment with backtracking on FPGAs. The proposed

scheme can efficiently reduce the storage cost by shortening the length of PE local memory, and increase the scale of PE array fitted in FPGA. Experimental results showed our implementation is superior to related works in frequency, maximum PE number and peak performance.

References

1. GenBank Growth Statistics (March 7 2006), http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html
2. Regester, K., Byun, J., et al.: Implementing Bio-informatics Algorithms on Nallatech-Configurable Multi-FPGA Syatems.University of North Carolina (2005)
3. Oliver, T., Schmidt, B., Maskell, D.: Hyper Customized Processors for Bio-Sequence Database Scanning on FPGAs. In: Proc. ACM/SIGDA 13th International Symposium on Field Programmable Gate Arrays, pp. 229–237 (2005)
4. Court, T.V., et al.: Families of FPGA-Based Accelerators for Approximate String Matching. Journal of Microprocessors and Microsystems 31, 135–145 (2007)
5. Dydel, S., Bala, P.: Large Scale Protein Sequence Alignment Using FPGA Reprogrammable Logic Devices. In: Proc. IEEE Int. Conf. Field Programmable Logic and Application, pp. 23–32. IEEE Computer Society Press, Los Alamitos (2004)
6. Yu, C.W., Kwong, K.H., et al.: A Smith-Waterman Systolic Cell. Proc. IEEE Int. Conf. Field Programmable Logic and Application, 375–384 (2003)
7. Peiheng, Z., Xinchun, L., Xiangyang, J.: An Implemention of Reconfigurable Computing Accelerator Card Oriented Bioinformatics. Journal of Computer Research and development, 930–937 (2005)
8. West, B., et al.: An FPGA-based Search Engine for Unstructured Database. In Proc. of 2nd Workshop on Application Specific Processors, 25–32 (2003)
9. Herbordt, M.C., Model, J., et al.: Single Pass, BLAST-Like, Approximate String Matching on FPGAs. In: Proc. IEEE 14th IEEE Int. Symp. Field-Programmable Custom Computing Machines, pp. 217–226 (2006)
10. Michailidis, P.D., Konstantinos, G.: Margaritis:A Programmable Array Processor Architecture for Flexible Approximate String Matching Algorithms. Journal of Parallel and Distributed Computing 67, 131–141 (2007)
11. El-Ghazawi, T.: The High-Performance Reconfigurable Computing Era. GWU HPC Symposium (2006)

Multi-cluster Load Balancing Based on Process Migration*

XiaoYing Wang, ZiYu Zhu, ZhiHui Du, and SanLi Li

Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science & Technology, Tsinghua University, Beijing 100084
{wangxy,zzy,duzh,lsl}@tirc.cs.tsinghua.edu.cn

Abstract. Load balancing is important for distributed computing systems to achieve maximum resource utilization, and process migration is an efficient way to dynamically balance the load among multiple nodes. Due to limited capacity of a single cluster, it's necessary to share the underutilized resources of other sites. This paper addresses the issues in multi-cluster load balancing based on process migration across separate clusters. Key technology and mechanisms are discussed and then the implementation of a prototype system is described in detail. Experimental results depict that by achieving multi-cluster load balance, surplus resources can be efficiently utilized and the makespan is also greatly reduced as a result.

Keywords: Load balancing, Process migration, Multi-cluster.

1 Introduction

Recent years, clusters of inexpensive networked computers are increasingly a popular platform for executing computationally intense and long running applications. The main goal of cluster systems is to share all the resources of the whole system through the interconnections and to effectively share resources via efficient resource management and task scheduling, finally achieving high performance. Thus, a key problem is how to effective utilize the resources. However, in such a loose-coupled computing environment as clusters, load imbalance, which is usually caused by the variable distribution of workload on the computing nodes, leads to performance degradation. Therefore, it is important for a cluster to balance the load among all nodes to improve the system performance.

Methods for load balancing can be classified into two categories. Static approaches are usually achieved by task allocation beforehand, which requires prior knowledge of exact information of tasks (such as execution time) and thus cannot adapt to the run-time load variation. Dynamic approaches are more complex and need the support of process migration, one of the most important techniques

* This work is supported partly by the Natural Science Foundation of China under Grant No.60503090 and No. 10778604, and China's National Fundamental Research 973 Program No.2004CB217903.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 100–110, 2007.

to fully utilize the resources of the entire system. Besides, process migration in a cluster system is also of benefit to system maintenance, availability, data locality, mobility and failure recovery [1,2,3]. Process migration can be implemented on different levels, which greatly affects the design choices. Current implementations can be divided into four categories:

- `Unix-like Kernel`. This method requires significant modification to monolithic kernel to achieve full transparency. Nevertheless, kernel-level implementation could sufficiently utilize the functionality of the operating systems(OS), obtain the detailed status about processes and thus provide high efficiency to users. Examples are LOCUS [5] and Sprite [6].
- `Microkernel`. As a separate module, microkernel builds process migration facilities on top of OS. It's relatively easy to implement because of the availability of system transparency support and message passing communication mechanism. Mach [7] is a typical example.
- `User-space migration`. This method doesn't require the modification of the kernel, implying that the entire address spaces are extracted and transferred to rebuild at the destination node. Condor [8] is an example system. Since not all states can be obtained at user level, some processes cannot be migrated. Great overhead has to be paid for breaking the obstacle between kernel space and user space, and thus the efficiency is much lower.
- `Application-level migration`. The function of process migration is integrated into real applications, and optimization methods could be designed based on particular applications. Freedman [9] and Skordos [10] have studied about such approaches. However, transparency and reusability would be significantly sacrificed, since semantics of the application need to be known and the programs have to be rewritten or recompiled. The functions of migrated processes are limited, and each new application has to be specially designed.

OpenMosix [4] is a typical tool which modifies the OS kernel to support process migration and allows multiple uniprocessors and symmetric multiprocessors running the same kernel to work in close cooperation. Dynamic load balancing can be achieved by migrating processes from one node to another, preemptively and transparently. Via openMosix, a cluster is seen as a large virtual machine with multiple CPUs and mass storage, and the cluster-wide performance of both sequential and parallel applications can be improved. However, when the capacity of a single cluster is limited, resources of multiple clusters ought to be shared. Unfortunately, conventional implementations of kernel-level process migration don't support migrating across different clusters. As computational grids emerged and got widely used, resources of multiple clusters became dominant computing nodes of the grid. Cross-cluster process migration can help to balance the load among multiple grid nodes with fine granularity, so it's also valuable for multi-site load balancing in computational grid. In this paper we discuss the key issues in multi-cluster load balancing and considerations in the implementation of a prototype system. Experimental results demonstrate that resource utilization greatly benefits from enabling multi-cluster load balancing, thus leading to significant reduction in task makespan.

2 Process Migration Techniques

In this section, we discuss the key techniques and mechanisms of process migration based on openMosix [11,12]. Each process has only one Unique Home Node(UHN) where it was created, usually the node which the user logged in. Every process seems to be running on its UHN, but in fact it may have been transferred to another node.

2.1 Deputy/Remote Mechanism

A migrated process consists of two parts - user context (called *remote*) and system context (called *deputy*). *Remote* contains the program code, stack, data, memory-maps and registers of the process, encapsulating the process when it's running at user level. Meanwhile, *deputy* encapsulates the process when it's running at kernel level, containing the description of resources that the process is attached to and a kernel-stack for the execution of system code on behalf of the process. Thus, *remote* can be migrated several times among different nodes, while *deputy* is site-dependent and can only stay on UHN. The interfaces between user context and system context are well defined. It's possible to intercept every interaction between the two contexts, and forward it across the network.

In Linux operating system, a process can only enter the kernel level via system calls. The interception and forwarding of every interaction between two contexts are done by the Link Layer. When a process has been migrated, its *deputy* and *remote* still keep connected. *Remote* deals with UHN-dependent operations through *deputy*, like getting environment variables or doing I/O. High transparency is achieved by their interaction. When *remote* meets system calls or resource requests when running the process, it sends them back to *deputy*. *Deputy* is always looping itself in kernel level, waiting for the requests from *remote*. Then, it deals with them and sends the results back to *remote*.

2.2 Migration Procedure

After the OS with migration support boots up, three daemons are started on each node running as kernel-level threads, including: *mig_daemon*, which listens at a certain port and deals with incoming requests; *info_daemon*, which collects and distributes the load information; and *mem_daemon*, which monitors memory utilization. Necessary steps of the process migration procedure include: (1) *Target selection*. A target node can be either specified by user manually or automatically decided by the scheduling system according to history and current load information of the cluster. When a process P is selected to migrate, it is marked and its priority is increased in order to get CPU timeslices more easily. Then, a request is sent to the *mig_daemon* on the target node. (2) *Request and negotiation*. After receiving the migration request of process P, the target node forks a new process P' to deal with it. P' first asks the load-balancing module whether to accept the incoming request and the module judges it according to predefined algorithms. P' is marked as remote, and then a TCP connection is established

between *remote* and *deputy*, exchanging messages between two nodes. (3)*Process state transfer*. If the target node accepts P to migrate there, it sends back an acknowledgement to the source node. After receiving the acknowledgment message, the source node starts to extract the states of process P and forwards them to the remote process. The *remote* modifies its own process states according to the received data. (4)*Execution resuming*. Once the sending phase is finished, P becomes *deputy* and enters a waiting loop in kernel level, until the process is terminated or migrated back to its UHN. P' is modified to *READY* state and added into the running queue, waiting for scheduling. When it is scheduled and enters the use space, it resumes executing from the instruction before migrating, which indicates the completion of the entire process migration procedure.

3 Multi-cluster Load Balancing Implementation

In reality, cluster systems often belong to different organizations or locate in isolated districts. Extending and merging the capability of existing cluster systems which are geographically distributed is helpful to achieve unified management and also decrease the cost of system expansion. In this section we present the main issues to implement a system supporting multi-cluster load balancing.

3.1 Multi-cluster Architecture

In a cluster system, computing nodes are usually configured inside a Local Area Network. A gateway node is connected to all the nodes via one Network Interface Card(NIC), and to the public network via another NIC, assuring that external users can log in and operate. For security consideration, users from public network cannot access the internal nodes of a cluster directly. The gateway node acts as a router, and also guarantees the security of the cluster system via firewall configurations. The topology of multi-cluster architecture is shown in Fig. 1.

As seen, the computing nodes inside different clusters are unable to communicate with each other. Traditional process migration approaches require direct connection of UHN and the remote node. As a result, when a cluster is under heavy load while other clusters are light-loaded, the imbalance problem emerges from a global point of view. To support multi-cluster load-balancing, lower-level operation system needs to be modified and the gateway can be used to forward migration data and necessary messages. Here, we add a property *groupID* into the data structure representing a node, indicating the cluster it belongs to. The gateway node is not involved in computing and has two entries in the configuration file - one for internal IP address, which contains a *groupID* of the cluster it belongs to; another for public IP address, which has a *groupID* of 0. An example file of two clusters is shown in Fig. 2.

3.2 Information Collection

There is no central control or master/slave relationship between nodes. For automatic load-balancing, every node has to know the current status of other nodes.

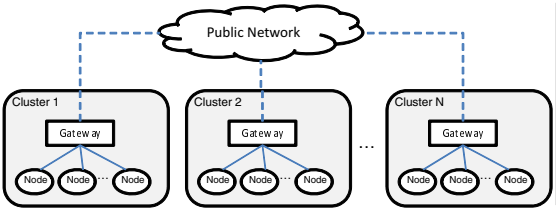


Fig. 1. Architecture of Multiple Clusters

```
# ID  IP  number-of-nodes groupID
# ================================
    1 10.0.0.1       1        1
    2 10.0.0.5       1        1
    3 10.0.0.6       gw       1
    3 192.168.4.15   gw       0
    4 192.168.4.11   gw       0
    4 10.0.0.111     gw       2
    5 10.0.0.9       1        2
    6 10.0.0.12      1        2
```

Fig. 2. Config File Example

This is achieved by each node sending its own info to others periodically. Hence, it's necessary to enable the support for information collection in multi-cluster scenario. Load information is forwarded using UDP protocol, as the information messages needn't acknowledgement and have no critical requirements for reliable transfer. Each node starts an *info_daemon*, waiting for info messages from other nodes. A node sends back an info message about itself immediately after receiving another's.

In order to let the gateway know where to send these messages, two additional segments are added, each carrying a *sockaddr* structure of the network address. The converting and sending procedure of the encapsulated data is shown in Fig. 3, which involves three types of nodes working: 1)*Sender(Src)*. The source node first queries the *groupID* of the destination node and compares to its own *groupID*. If unequal, then they are not in a same cluster and cross-cluster migration is needed. Then, it queries the configuration table again to get the external address of the gateway node of the destination cluster together with the internal address of the gateway node of the local cluster. Data are encapsulated as a three-segment format, with the first segment containing the destination gateway address(*Gw2 addr*), the second segment containing the destination node address (*Dest addr*), and the last segment containing the original message content. The encapsulated data are sent to the local gateway. 2)*Gateway node(Gw)*. The gateway is listening and waiting for the encapsulated data from *Src*. It doesn't care the content, only processing and converting messages. It reads the first segment and extracts the network address as the next destination, then copies the second segment to the first segment, and fills the second segment with the address of the original sender. 3)*Receiver(Dest)*. After converted by two gateway nodes sequentially, the first segment of the encapsulated data becomes the address of *Src* and the second segment is the gateway of the source cluster(*Gw1*). The receiver can query the configuration table and find the *ID* of *Src*, and then obtain load info about *Src* from the last segment of received data.

3.3 Cross-Cluster Process Migration

According to the source and destination of the process to be migrated, we describe following three different scenarios.

Local to remote (migrating to a remote node). Before migrating, the local node first connects to its *mig_daemon* and sends requests containing the reason to migrate. After acknowledged by the remote node, necessary data are transferred

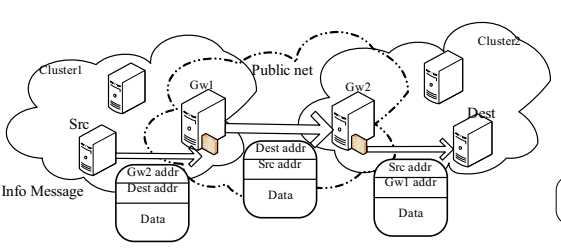


Fig. 3. Info Messages Forwarding

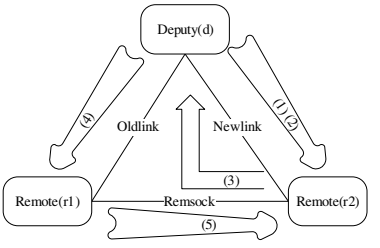


Fig. 4. remote to remote

continuously, including process memory states, virtual areas, physical pages and so forth. Once finishing, the local process becomes deputy and enters a loop waiting for system-calls from remote. At the remote node, *mig_daemon* accepts incoming migration requests and sends back acknowledgement if it agrees to receive the process. Then, a user-level process is created and after receiving all necessary data, the process is changed to *READY* state and waits for scheduling.

Remote to local(migrating back home). When a process is coming back from the remote node, the UHN sends a *DEP_COME_BACK* instruction, and then receives all of the process states. Remote starts "going home" after being acknowledged by UHN. After finishing sending the necessary data, remote kills the migrated process and exits normally.

Remote to remote(migrating more than once). It's a little more complex when a process needs to migrate from one remote node to another. Figure 4 illustrates the steps of migration from remote *r1* to *r2*. (1)*oldlink* is the link between *d* and *r1*, and *newlink* connecting *d* and *r2* is created. (2) *d* sends a probing request to *r2*, and notifies it a migration event is going to occur. (3) *r2* sends information of itself to *d*. (4) *d* copies the information about *r2* and sends it together with a request to *r1* via *oldlink* and tells it "Please Migrate". (5) *r1* opens a new link to *r2*, and then sends a migration request to *r2*. After sending necessary data to *r2*, it releases the local process. Till now, the process is migrated to *r2* and only related to *d* and *r2*.

In order to transfer process states reliable TCP connections should be established among nodes and gateways. Unlike info messages, migration messages need bidirectional transfer. The source node sends an "open-connection" command first, the structure of which is similar to the packed info message, with the third segment containing a command string. Since the *mig_gateway* acts almost in the same way as the *info_gateway*, detailed descriptions are omitted here. Note that if the migration type is "remote to remote", the listening port will be randomly selected. Once the process terminates, the source node notifies the gateway to close all open connections.

3.4 Load Balancing Strategy

The main load balancing algorithms consists of CPU load-balancing and memory ushering. Due to space constraints, here we only focus on the CPU load-balancing

strategy. The dynamic CPU load-balancing algorithm continuously attempts to reduce the load differences between pairs of nodes, by migrating processes from high loaded to less loaded nodes. This scheme is decentralized with all the nodes regarding each other as equal-position peers and executing the same algorithm. The whole balancing strategy is comprised of following steps.

1)Calculate the load values. Since the nodes in the system may be heterogeneous, the load calculation is related to the number of processors(denoted as N_p) and their speed. First, the number of running processes is added to an accumulator at each clock interrupt. Then, the accumulated load L_{acc} is normalized to the CPU speed of this processor using the maximum CPU speed S_{MAX} in the system versus the local node's calculated CPU speed S_{cal}, namely,

$$L_{acc} = L_{acc} \cdot \frac{S_{MAX}}{S_{cal} \cdot N_p}. \tag{1}$$

Under the consideration of preventing migration thrashing, the actual load value sent to other nodes, called "the export load", is calculated to be slightly higher than the load calculated for internal use. Denote the internal load as L_{int} and the export load as L_{exp}. To calculate L_{exp}, a value representing the highest load over any period has to be recorded, denoted as L_{up}. Denote L_{income} as the summarized load brought by processes"recently" migrated to this node. Then, the internal load and the export load can be computed in the following way:

```
If (L_acc>L_int) //slowly up
    L_int=L_int*decay+L_acc
Else //quickly down
    L_int=L_acc
End If
If (L_acc>=L_up) //quickly up
    L_up=L_acc
Else //slowly down
    L_up=(L_up*7+L_acc)/8
End If
L_exp=L_up+L_income //prevent receiving too many new processes too quickly
```

where *decay* is predefined constant value between 0∼1. Each node maintains a local load vector storing both the internal load value of its own and export load values received from other nodes.

2)Choose a process to migrate. Processes cannot be migrated if they are constrained, such as being locked, in creation phase, being transferred, or using a shared memory region. Moreover, if a process has not accumulated enough CPU usage, it is not considered for migration. For each process, a migration priority value is first calculated based on the CPU use since it is last considered for migration, including the CPU use of all its children processes. Then, this priority value is combined with a value which attempts to measure the process's contribution to the load currently on the machine. Consequently, a process which forks frequently is more attractive for migration, because once migrated it will continue to fork children thus spreading the load as it bounces from one node to another. Once a process is chosen to be migrated, a flag will

be set, indicating that it's the candidate process selected for load-balancing consideration.

3)Choose a destination node. It is a complicated problem to determine the optimal location for the job to be migrated, since available resources are usually heterogeneous and even not measured in the same units. Here we try to reconcile these differences by standardizing the resource measurements and adopt a method based on economic principles and competitive analysis. The target node for migration is determined by computing the opportunity cost for each of the nodes in the local load vector, which is a concept from the field of economies research. The key idea is to convert the total usage of several heterogeneous resources, such as memory and CPU, into a single homogeneous "cost". Jobs are then assigned to the machine where they have the lowest cost, just like in a market oriented economy. A simple way is to compute the marginal cost of a candidate node, namely, the amount of the sum of relative CPU usage and memory usage would increase if the process was migrated to that node. The goal is to find a node with minimal marginal cost and select it as the destination.

As a whole, the above load-balancing algorithms respond to variation in the runtime characteristics of the processes, as long as there is no extreme shortage of other resources such as free memory or empty process slots.

4 Performance Evaluation

This section presents the results of performance evaluation experiments conducted on the prototype system we implemented. Our testbed is comprised of two clusters: *Cluster 1* has nodes with dual PIII 500MHz CPUs and 128M physical memory; *Cluster 2* has nodes with dual PIII 733MHz and 256M physical memory. There are two independent networks of each cluster, and they own a gateway node respectively, connecting to the public net. Gateways can communicate to each other directly. Both nodes inside the two clusters and the gateways are connected by 100Mb/s Ethernet. After the startup of our prototype system on each node, all the load information used by the scheduling module can be monitored by upper-level tools.

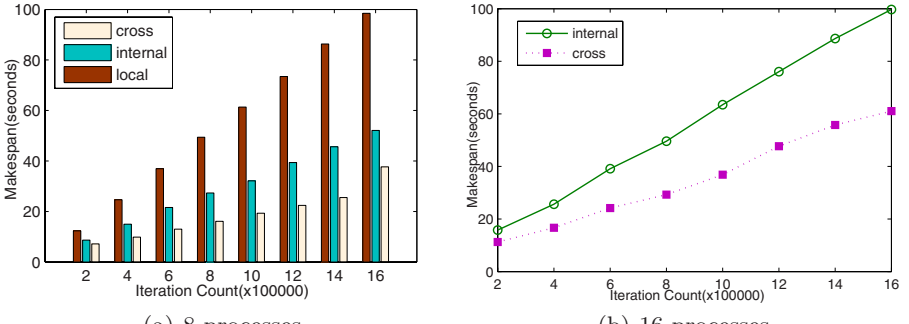
We use a simple CPU-intensive program for exemplification of the experiment, which requires two parameters - the iteration count and the number of child processes. As iteration count becomes larger, the CPU load will be accordingly heavier. We tested and compared three scenarios respectively: without migration support (*local*), with intra-cluster migration support (*internal*) and with cross-cluster migration support (*cross*). Figure 5 shows the task completion time (i.e. makespan) achieved in the above three scenarios, in which Fig. 5(a) has 8 child processes forked and Fig. 5(b) has 16 ones.

During the period when the program is running, we can monitor the load situation on the nodes by user-level tools. While the operating system treats all processes as a whole task, through the userspace monitoring tool implemented in our prototype system, we can distinguish every child process clearly. Take Fig. 5(b) for example, when the program has been submitted to a node and

starts to run, the load of the local cluster quickly increases, while the other cluster is idle without any heavy-load tasks; after enabling multi-cluster load balancing, 4 processes are distributed on each node, and the CPU utilization of all nodes reaches nearly 100%.

From Fig. 5, it can be observed that without migration support, it results in low efficiency and remarkably takes more time to complete the task. If the intra-cluster migration support is enabled, a half time can be saved (because there are two nodes in the cluster to share the load), but nodes inside the cluster are still under heavy burden. With multi-cluster load balancing, load can be shared across different clusters and thus the task makespan is greatly reduced. The speedup is computed and shown in Table 1, which depicts the significant improvement by enabling multi-cluster load balancing. Moreover, from the table we can also observe that as the iteration count becomes larger, the speedup of *cross* increases proportionally. That's because in this program the iteration count represents the computation amount of the tasks, and long-running tasks are more tolerant of "non-computation" overhead because of their urgent requirement for additional resource. As the results demonstrate, multi-cluster load balancing can make the resource of multiple clusters more effectively utilized and workload more balanced in a total view.

Nevertheless, the performance overhead cannot be ignored, especially when the migration experiences two hops at the gateway nodes. Now we investigate the multi-cluster load balancing overhead, which involves information analysis and making scheduling decisions, process states transfer, communication over network, and results finalization. We figure out the average summarized overhead per computing node and compared *cross* with *internal* in order to make clear how much additional overhead is caused by the processing of gateway nodes, as shown in the bottom rows of Table 1. Again we can see that larger amount of computation makes the overhead relatively smaller, as explained previously. Meanwhile, as the computation amount increases, the additional overhead incurred by *cross*(relative to *internal*) becomes more significant too, due to massively more data passing through the intermediate gateway nodes. However, to sum up, *cross* only incur less than 3% more overhead compared to the inter-



(a) 8 processes (b) 16 processes

Fig. 5. Performance Evaluation Results

Table 1. Comparison of Speedup and Overhead

Iteration($\times 10^5$)		2	4	6	8	10	12	14
SpeedUp	Internal	1.43	1.65	1.71	1.81	1.91	1.86	1.89
	Cross	1.72	2.51	2.84	3.07	3.17	3.28	3.38
Average Overhead Per Node	Internal	14%	8.8%	7.1%	4.8%	2.3%	3.4%	2.7%
	Cross	14%	9.3%	7.2%	5.8%	5.2%	4.5%	3.9%

nal load balancing, which is acceptable especially when dealing with long-term tasks. In practice, the upper-level scheduler should be able to decide whether to migrate according to the overhead and possible efficiency gain.

5 Conclusions and Future Work

In this paper, we conduct researches on multi-cluster load-balancing based on the study and analysis of process migration mechanism. The main problem is that the internal node of a cluster cannot be directly accessed by other machines outside the cluster. Hence, we have designed a system which supports multi-cluster load balancing by employing gateway nodes to forward and transfer necessary data. The results of performance evaluation experiments based on the prototype system have demonstrated the availability and efficiency of multi-cluster load balancing in reducing the task makespan. Our work can be regarded as a preliminary step into the research on dynamical resource sharing and load balancing of multiple large nodes in computational grid environment (especially for CPU-intensive applications). This prototype system has been put into practice and achieved unified management and resource sharing among multiple clusters. As a possible direction for future work, we plan to investigate some potential problems, such as the bottleneck of gateway nodes when dealing with frequent system-calls. Moreover, in the case of decentralized grid environment, the latency between gateway nodes may be considerable, which requires the decision of whether and when there exists the need to migrate. As a further step, we are also considering to employ modeling and simulation to test the performance of multiple large cluster systems in the grid under different workloads.

References

1. Milojicic, D.S., Douglis, F., Paindaveine, Y., et al.: Process migration. ACM Comput. Surv. 32(3), 241–299 (2000)
2. Powell, M.L., Miller, B.P.: Process migration in DEMOS /MP. In: Proc. Ninth Symposium on Operating System Principles, pp. 110–119. ACM, New York (1983)
3. Smith, P., Hutchinson, N.C.: Heterogeneous process migration: The tui system. Technical Report TR-96-04, University of British Columbia. Computer Science (1996)
4. The openMosix Project, http://openmosix.sourceforge.net
5. Popek, G.J., Walker, B.J., Johanna, M., et al.: LOCUS - A Network Transparent, High Reliability Distributed System. Proceedings of the 8th Symposium on Operating System Principles, 169–177 (1981)

6. Douglis, F., Ousterhout, J.K.: Transparent Process Migration: Design Alternatives and the Sprite Implementation. Softw., Pract. Exper. 21(8), 757–785 (1991)
7. Accetta, M., Baron, R., Bolosky, W., et al.: Mach: A New Kernel Foundation for UNIX Development. In: Proceedings of the Summer USENIX Conference, pp. 93-112 (1986)
8. Litzkow, M., Solomon, M.: Supporting Checkpointing and Process Migration outside the UNIX Kernel. Proceedings of the USENIX Winter Conference, pp. 283-290 (1992)
9. Freedman, D.: Experience Building a Process Migration Subsystem for UNIX. In: Proceedings of the WinterUSENIX Conference, pp. 349-355 (1991)
10. Skordos, P.: Parallel Simulation of Subsonic Fluid Dynamics on a Cluster of Workstations. In: Proceedings of the Fourth IEEE International Symposium on High Performance Distributed Computing (1995)
11. Argentini, G.: Use of openMosix for parallel I/O balancing on storage in Linux cluster. CoRR cs.DC/0212006 (2002)
12. Katsubo, D.: Using openMosix Clustering System for Building a Distributed Computing Environment, http://www.openmosix.org.ru/docs/omosix.html

Property-Preserving Composition of Distributed System Components

K.S. Cheung[1] and K.O. Chow[2]

[1] Hong Kong Baptist University, Kowloon Tong, Hong Kong
cheungks@hkbu.edu.hk
[2] City University of Hong Kong, Tat Chee Avenue, Hong Kong
cspchow@cityu.edu.hk

Abstract. Augmented marked graphs possess a special structure for modelling common resources as well as some desirable properties pertaining to liveness, boundedness, reversibility and conservativeness. This paper investigates the property-preserving composition of augmented marked graphs for the synthesis of distributed systems. It is proposed that distributed system components are specified as augmented marked graphs. An integrated system is then obtained by composing these augmented marked graphs via their common resource places. Based on the preservation of properties, the liveness, boundedness, reversibility and conservativeness of the integrated system can be readily derived. This effectively solves the difficult problem of ensuring design correctness in the composition of distributed system components.

1 Introduction

In the past decade, component-based system design has emerged as a promising paradigm to meet the ever increasing needs for managing system complexity and maximising re-use as well as for deriving software engineering into standards. When applied to distributed systems which usually involve concurrent (parallel) and asynchronous processes, one need to be aware that errors such as deadlock and capacity overflow may occur. Even though the system components are correct in the sense that they are live (implying freeness of deadlock), bounded (implying absence of capacity overflow) and reversible (implying the capability of being reinitialised from any reachable states), the integrated system may not be correct, especially as competition of common resources exists.

This paper investigates the component-based approach to synthesising a given set of distributed system components into an integrated system. Our focus is placed on the preservation of four essential properties which include liveness, boundedness, reversibility and conservativeness. Based on the property-preserving composition of augmented marked graphs, we propose a formal method for synthesising the given distributed system components into an integrated system whose design correctness (in terms of liveness, boundedness, reversibility and conservativeness) can be readily derived and verified.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 111–120, 2007.

A subclass of Petri nets, augmented marked graphs possess a special structure for modelling common resources. They exhibit some desirable properties pertaining to liveness, boundedness, reversibility and conservativeness. Chu and Xie first studied their liveness and reversibility using siphons and mathematical programming [1]. We proposed siphon-based and cycle-based characterisations for live and reversible augmented marked graphs, and transform-based characterisations for bounded and conservative augmented marked graphs [2, 3, 4]. Besides, the composition of augmented marked graphs via common resource places was preliminarily studied [5, 6].

In this paper, after a brief review of augmented marked graphs, we investigate the composition of augmented marked graphs via common resource places and show that this composition preserves boundedness and conservativeness whereas liveness and reversibility can be preserved under a pretty simple condition. The results are then applied to the composition of distributed system components, where liveness, boundedness, reversibility and conservativeness of the integrated system can be readily derived. These will be illustrated using examples.

The rest of this paper is organised as follows. Section 2 introduces augmented marked graphs. Section 3 presents the composition of augmented marked graphs with a special focus on the preservation of properties. Section 4 shows its application to the composition of distributed system components. Section 5 briefly concludes this paper. Readers of this paper are expected to have knowledge of Petri nets [7, 8].

2 Augmented Marked Graphs

This section introduces augmented marked graphs and summarises their known properties and characterisations.

Definition 2.1 [1]**.** An augmented marked graph $(N, M_0; R)$ is a PT-net (N, M_0) with a specific subset of places R called resource places, satisfying the following conditions : (a) Every place in R is marked by M_0. (b) The net (N', M_0') obtained from $(N, M_0; R)$ by removing the places in R and their associated arcs is a marked graph. (c) For each $r \in R$, there exist $k_r \geq 1$ pairs of transitions $D_r = \{ \langle t_{s1}, t_{h1} \rangle, \langle t_{s2}, t_{h2} \rangle, ..., \langle t_{skr}, t_{hkr} \rangle \}$ such that $r^\bullet = \{ t_{s1}, t_{s2}, ..., t_{skr} \} \subseteq T$ and $^\bullet r = \{ t_{h1}, t_{h2}, ..., t_{hkr} \} \subseteq T$ and that, for each $\langle t_{si}, t_{hi} \rangle \in D_r$, there exists in N' an elementary path ρ_{ri} connecting t_{si} to t_{hi}. (d) In (N', M_0'), every cycle is marked and no ρ_{ri} is marked.

Definition 2.2. For a PT-net (N, M_0), a set of places S is called a siphon if and only if $^\bullet S \subseteq S^\bullet$. S is said to be minimal if and only if there does not exist a siphon S' in N such that $S' \subset S$. S is said to be empty at a marking $M \in [M_0\rangle$ if and only if S contains no places marked by M.

Definition 2.3. For a PT-net (N, M_0), a set of places Q is called a trap if and only if $Q^\bullet \subseteq {}^\bullet Q$. Q is said to be maximal if and only if there does not exist a trap Q' in N such that $Q \subset Q'$. Q is said to be marked at a marking $M \in [M_0\rangle$ if and only if Q contains a place marked by M.

Property 2.1 [1]**.** An augmented marked graph is live and reversible if and only if it does not contain any potential deadlock. (Note : A potential deadlock is a siphon which would eventually become empty.)

Definition 2.4. For an augmented marked graph (N, M_0; R), a minimal siphon is called a R-siphon if and only if it contains at least one place in R.

Property 2.2 [1, 2, 3]**.** An augmented marked graph (N, M_0; R) is live and reversible if every R-siphon contains a marked trap.

Property 2.3 [2, 3]**.** An augmented marked graph (N, M_0; R) is live and reversible if and only if no R-siphons eventually become empty.

Definition 2.5 [4]**.** Suppose an augmented marked graph (N, M_0; R) is transformed into a PT-net (N', M_0') : For each r ∈ R, where D_r = { ⟨t_{s1}, t_{h1}⟩, ⟨t_{s2}, t_{h2}⟩, ..., ⟨t_{skr}, t_{hkr}⟩ }, replace r with a set of places { q_1, q_2, ..., q_{kr} } such that $M_0'[q_i] = M_0[r]$ and $q_i^\bullet$ = { t_{si} } and ${}^\bullet q_i$ = { t_{hi} } for i = 1, 2, ..., k_r. (N', M_0') is called the R-transform of (N, M_0; R).

Property 2.4 [4]**.** Augmented marked graph (N, M_0; R) is bounded and conservative if and only if every place in its R-transform (N', M_0') belongs to a cycle.

Fig. 1 shows an augmented marked graph (N, M_0; R), where R = { r_1, r_2 }. Every R-siphon contains a marked trap and would never become empty. It follows from Properties 2.2 and 2.3 that (N, M_0; R) is live and reversible. As every place in the R-transform of (N, M_0; R) belongs to a cycle, according to Property 2.4, (N, M_0; R) is bounded and conservative.

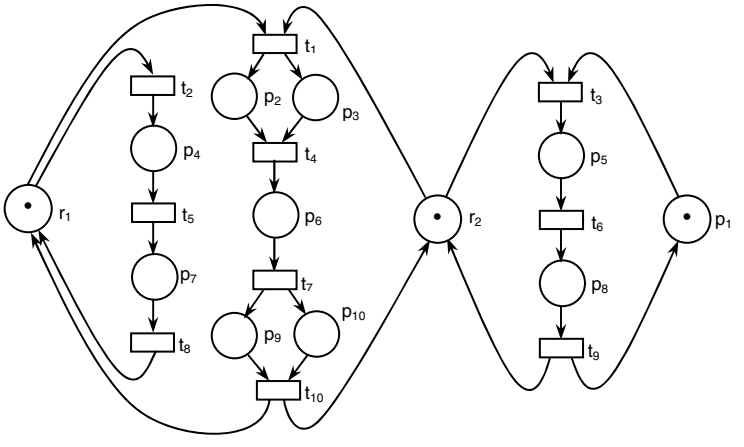


Fig. 1. An augmented marked graph

3 Composition of Augmented Marked Graphs

This section first describes the composition of augmented marked graphs via common resource places. Preservation of properties is then studied.

Property 3.1. Let (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) be two augmented marked graphs, where R_1' = { r_{11}, r_{12}, ..., r_{1k} } ∈ R_1 and R_2' = { r_{21}, r_{22}, ..., r_{2k} } ∈ R_2 are the common places that r_{11} and r_{21} are to be fused as one single place r_1, r_{12} and r_{22} into r_2, ..., r_{1k}

and r_{2k} into r_k. Then, the resulting net is also an augmented marked graph (N, M_0; R), where $R = (R_1 \setminus R_1') \cup (R_2 \setminus R_2') \cup \{ r_1, r_2, ..., r_k \}$. (obvious)

Definition 3.1. With reference to Property 3.1, (N, M_0; R) is called the composite augmented marked graph of (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) via a set of common resource places $\{ (r_{11}, r_{21}), (r_{12}, r_{22}), ..., (r_{1k}, r_{2k}) \}$, where $r_{11}, r_{12}, ..., r_{1k} \in R_1$ and $r_{21}, r_{22}, ..., r_{2k} \in R_2$. $R_F = \{ r_1, r_2, ..., r_k \}$ is called the set of fused resource places that are obtained after fusing $(r_{11}, r_{21}), (r_{12}, r_{22}), ..., (r_{1k}, r_{2k})$.

Fig. 2 shows two augmented marked graphs (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2). Fig. 3 shows the composite augmented marked graph (N, M_0; R) of (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) via $\{ (r_{11}, r_{21}) \}$, where $R_F = \{ r_1, r_2 \}$.

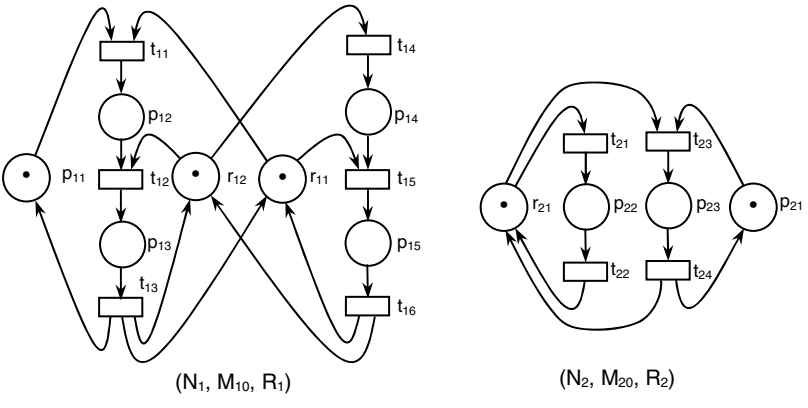


Fig. 2. Two augmented marked graphs (N_1, M_{10}, R_1) and (N_2, M_{20}, R_2)

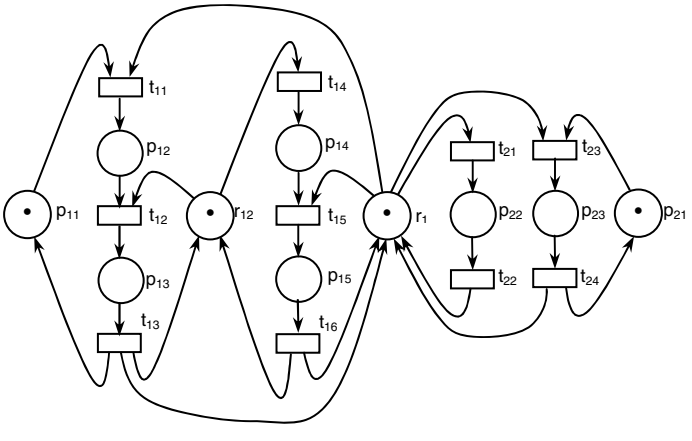


Fig. 3. An augmented marked graph obtained by composing the two augmented marked graphs in Fig. 2 via $\{ (r_{11}, r_{21}) \}$

Property 3.2 [5, 6]**.** Let (N, M_0; R) be the composite augmented marked graph of two augmented marked graphs (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) via a set of common resource places. (N, M_0; R) is bounded if and only if (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) are bounded.

Property 3.3 [5]**.** Let (N, M_0; R) be the composite augmented marked graph of two augmented marked graphs (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) via a set of common resource places. (N, M_0; R) is conservative if and only if (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) are conservative.

Definition 3.2. Let (N, M_0; R) be the composite augmented marked graph of two augmented marked graphs via a set of common resource places, and $R_F \subseteq R$ be the set of fused resource places. For (N, M_0; R), a minimal siphon is called a R_F-siphon if and only if it contains at least one place in R_F.

Property 3.4 [5]**.** Let (N, M_0; R) be the composite marked graph of two augmented marked graphs (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) via a set of common resource places. (N, M_0; R) is live and reversible if and only if (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) are live and no R_F-siphons eventually become empty.

Consider the augmented marked graphs (N_1, M_{10}; R_1) and (N_2, M_{20}; R_2) in Fig. 2. (N_1, M_{10}; R_1) is neither live nor reversible but is bounded and conservative. (N_2, M_{20}; R_2) is live, bounded, reversible and conservative. According to Properties 3.2 and 3.3, the composite augmented marked graph (N, M_0; R) as shown in Fig. 3 is bounded and conservative. According to Property 3.4, (N, M_0; R) is neither live nor reversible.

4 Application to Distributed Systems

In component-based system design, a system is synthesised from a set of components [9, 10]. It may not be live, bounded and reversible even all its components are live, bounded and reversible. For distributed systems which usually involve concurrent (parallel) and asynchronous processes, because of competition of common resources, errors such as deadlock and capacity overflow are easily induced. This section shows the application of composition of augmented marked graphs to the synthesis of a distributed system whose design correctness can be readily derived.

Fig. 4 shows a distributed system consisting of four system components, C_1, C_2, C_3 and C_4. Owing to the "distributed processing" nature, the components exhibit

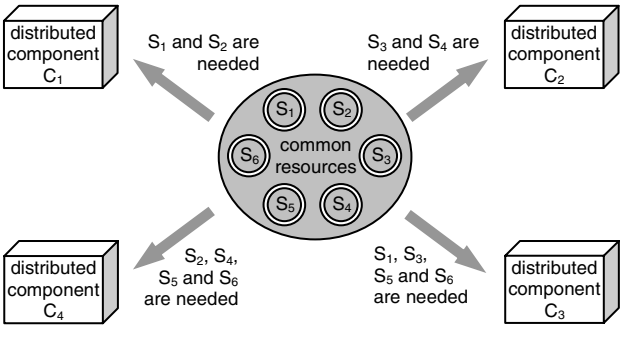


Fig. 4. Example of a distributed system with shared resources

concurrent (parallel) and asynchronous processes. There are six pieces of common resources, S_1, S_2, S_3, S_4, S_5 and S_6, used to be shared among the components.

The functions of the distributed system components C_1, C_2, C_3 and C_4 are briefly described as follows.

C_1 : At its initial idle state, C_1 invokes operation o_{11} only if S_1 is available. While o_{11} is being processed, S_1 is occupied. Once o_{11} finishes processing, operation o_{12} is invoked only if S_2 is available. S_1 is then released. While o_{12} is being processed, S_2 is occupied. Once o_{12} finishes processing, S_2 is released and C_1 returns to idle state. At any moment, S_1 is withheld on receipt of signal m_{11} and released on receipt of signal m_{12}. S_2 is withheld on receipt of signal m_{13} and released on receipt of signal m_{14}.

C_2 : At its initial idle state, C_2 invokes operation o_{21} only if S_3 is available. While o_{21} is being processed, S_3 is occupied. Once o_{21} finishes processing, operation o_{22} is invoked only if S_4 is available. S_3 is then released. While o_{22} is being processed, S_4 is occupied. Once o_{22} finishes processing, S_4 is released and C_2 returns to idle state. At any moment, S_3 is withheld on receipt of signal m_{21} and released on receipt of signal m_{22}. S_4 is withheld on receipt of signal m_{23} and released on receipt of signal m_{24}.

C_3 : At its initial idle state, C_3 invokes operation o_{31} only if S_1, S_3, S_5 and S_6 are all available. While o_{31} is being processed, S_1, S_3, S_5 and S_6 are occupied. Once o_{31} finishes processing, S_1, S_3, S_5 and S_6 are released and C_3 returns to idle state.

C_4 : At its initial idle state, C_4 invokes operation o_{41} only if S_2, S_4, S_5 and S_6 are all available. While o_{41} is being processed, S_2, S_4, S_5 and S_6 are occupied. Once o_{41} finishes processing, S_2, S_4, S_5 and S_6 are released and C_4 returns to idle state.

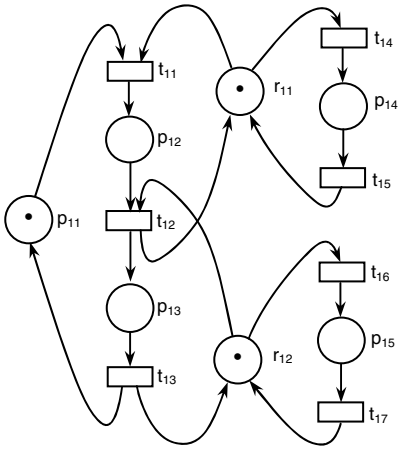
Our method begins with specifying each component as an augmented marked graph. We identify the event occurrences and their pre-conditions and post-conditions in the component. For each event occurrence, a transition is created for denoting the location of occurrence. Input and output places are created to denote the locations of its pre-conditions and post-conditions. An initial marking is created to denote the system initial state. Execution for the component begins at this initial marking which semantically means its initial idle state, and ends at the same marking.

Component C_1 is specified as augmented marked graph (N_1, M_{10}; R_1), where $R_1 = \{ r_{11}, r_{12} \}$. C_2 is specified as (N_2, M_{20}; R_2), where $R_2 = \{ r_{21}, r_{22} \}$. C_3 is specified as (N_3, M_{30}; R_3), where $R_3 = \{ r_{31}, r_{32}, r_{33}, r_{34} \}$. C_4 is specified as (N_4, M_{40}; R_4), where $R_4 = \{ r_{41}, r_{42}, r_{43}, r_{44} \}$. They are shown in Fig. 5.

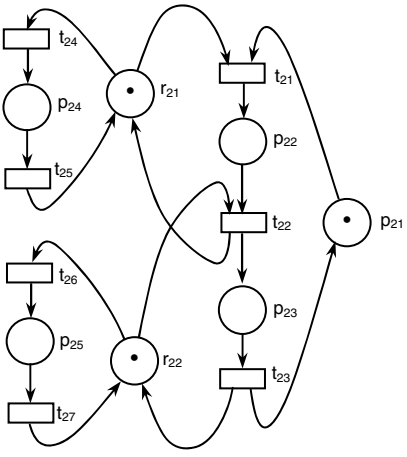
According to Properties 2.1, 2,2, 2.3 and 2.4, (N_1, M_{10}; R_1), (N_2, M_{20}; R_2), (N_3, M_{30}; R_3) and (N_4, M_{40}; R_4) are live, bounded, reversible and conservative.

Resource places r_{11} in (N_1, M_{10}; R_1) and r_{31} in (N_3, M_{30}; R_3) refer to the same resource S_1. r_{12} in (N_1, M_{10}; R_1) and r_{42} in (N_4, M_{40}; R_4) refer to the same resource S_2. r_{21} in (N_2, M_{20}; R_2) and r_{33} in (N_3, M_{30}; R_3) refer to the same resource S_3. r_{22} in (N_2, M_{20}; R_2) and r_{44} in (N_4, M_{40}; R_4) refer to the same resource S_4. r_{32} in (N_3, M_{30}; R_3) and r_{41} in (N_4, M_{40}; R_4) refer to the same resource S_5. r_{34} in (N_3, M_{30}; R_3) and r_{43} in (N_4, M_{40}; R_4) refer to the same resource S_6. (N_1, M_{10}; R_1), (N_2, M_{20}; R_2), (N_3, M_{30}; R_3) and (N_4, M_{40}; R_4) are to be composed via these common resource places.

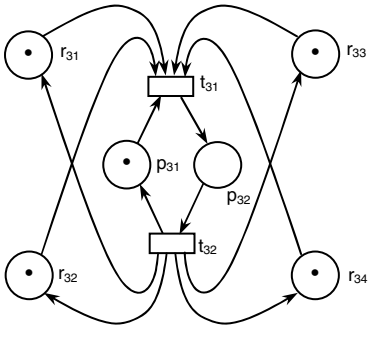
We first obtain the composite augmented marked graphs (N', M_0'; R') of (N_1, M_{10}; R_1) and (N_3, M_{30}; R_3) via { (r_{11}, r_{31}) }, and the composite augmented marked graph (N", M_0"; R") of (N_2, M_{20}; R_2) and (N_4, M_{40}; R_4) via { (r_{22}, r_{44}) }. Fig. 6 shows (N', M_0'; R'), where r_1 is the place after fusing r_{11} and r_{31}. Fig. 7 shows (N", M_0"; R"), where r_4 is the place after fusing r_{22} and r_{44}.



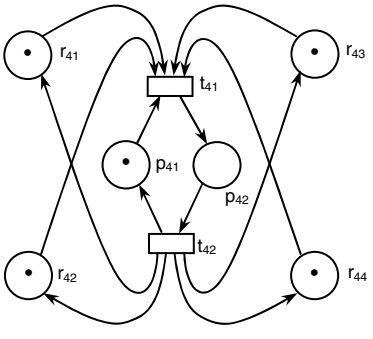
(N_1, M_{10}, R_1)



(N_2, M_{20}, R_2)



(N_3, M_{30}, R_3)



(N_4, M_{40}, R_4)

Semantic meaning of places

p_{11}	C_1 is at idle state
p_{12}	C_1 is performing operation o_{11}
p_{13}	C_1 is performing operation o_{12}
p_{14}	S_1 is being withheld
p_{15}	S_2 is being withheld
p_{21}	C_2 is at idle state
p_{22}	C_2 is performing operation o_{21}
p_{23}	C_2 is performing operation o_{22}
p_{24}	S_3 is being withheld
p_{25}	S_4 is being withheld
p_{31}	C_3 is at idle state
p_{32}	C_3 is performing operation o_{31}
p_{41}	C_4 is at idle state
p_{42}	C_4 is performing operation o_{41}
r_{11}, r_{31}	S_1 is available
r_{12}, r_{42}	S_2 is available
r_{21}, r_{33}	S_3 is available
r_{22}, r_{44}	S_4 is available
r_{32}, r_{41}	S_5 is available
r_{34}, r_{43}	S_6 is available

Semantic meaning of transitions

t_{11}	C_1 starts operation o_{11}
t_{12}	C_1 finishes operation o_{11} and starts operation o_{12}
t_{13}	C_1 finishes operation o_{12}
t_{14}	C_1 receives signal m_{11}
t_{15}	C_1 receives signal m_{12}
t_{16}	C_1 receives signal m_{13}
t_{17}	C_1 receives signal m_{14}
t_{21}	C_2 starts operation o_{21}
t_{22}	C_2 finishes operation o_{21} and starts operation o_{22}
t_{23}	C_2 finishes operation o_{22}
t_{24}	C_2 receives signal m_{21}
t_{25}	C_2 receives signal m_{22}
t_{26}	C_2 receives signal m_{23}
t_{27}	C_2 receives signal m_{24}
t_{31}	C_3 starts operation o_{31}
t_{32}	C_3 finishes operation o_{31}
t_{41}	C_4 starts operation o_{41}
t_{42}	C_4 finishes operation o_{41}

Fig. 5. Specification of distributed system components as augmented marked graphs

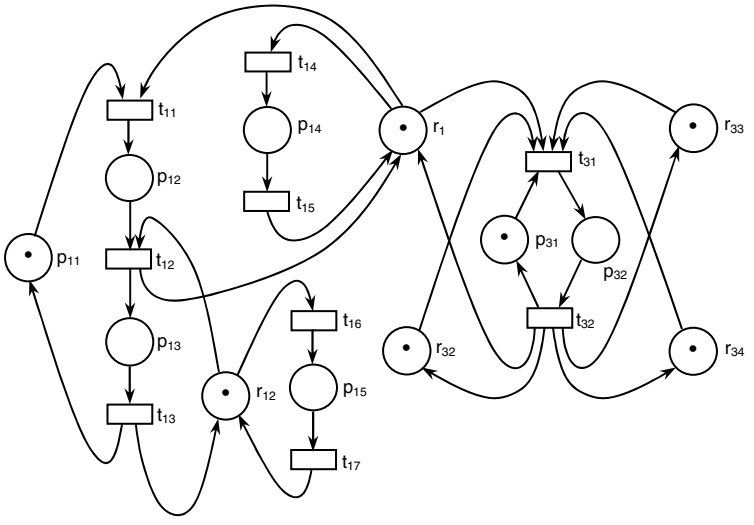


Fig. 6. Composite augmented marked graph (N', M_0'; R')

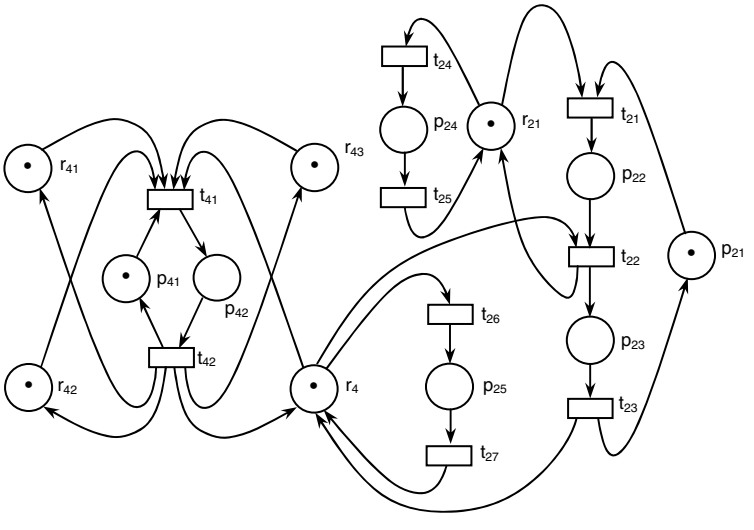
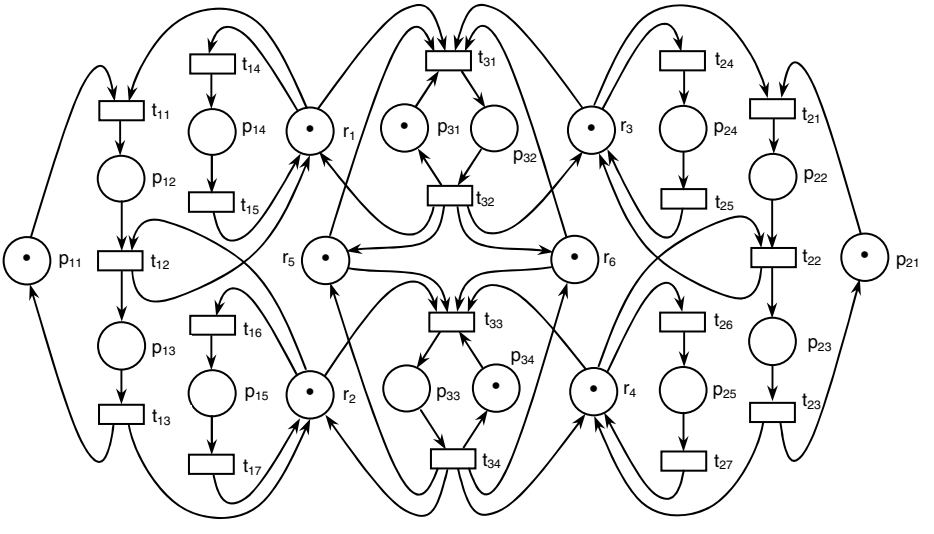


Fig. 7. Composite augmented marked graph (N", M_0"; R")

Since (N_1, M_{10}; R_1), (N_2, M_{20}; R_2), (N_3, M_{30}; R_3) and (N_4, M_{40}; R_4) are all bounded and conservative, according to Properties 3.2 and 3.3, the composite augmented marked graphs (N', M_0'; R') and (N", M_0"; R") are also bounded and conservative. On the other hand, (N_1, M_{10}; R_1), (N_2, M_{20}; R_2), (N_3, M_{30}; R_3) and (N_4, M_{40}; R_4) are all live and reversible. For (N', M_0', R'), where R_F' = { r_1 }, no R_F'-siphons would eventually become empty. According to Property 3.4, (N', M_0', R') is also live and reversible. For (N", M_0", R"), where R_F" = { r_4 }, no R_F"-siphons would eventually become empty. According to Property 3.4, (N", M_0", R") is also live and reversible.

We obtain the final composite augmented marked graph (N, M_0; R) of (N', M_0'; R') and (N", M_0"; R") via { (r_{12}, r_{42}), (r_{33}, r_{21}), (r_{32}, r_{41}), (r_{34}, r_{43}) }. Fig. 8 shows (N, M_0; R), where r_2 is the place after fusing r_{12} and r_{42}, r_3 is the place after fusing r_{21} and r_{33}, r_5 is the place after fusing r_{32} and r_{41}, and r_6 is the place after fusing r_{34} and r_{43}.

Since (N', M_0'; R') and (N", M_0"; R") are bounded and conservative, according to Properties 3.2 and 3.3, the composite augmented marked graph (N, M_0; R) is also bounded and conservative. On the other hand, (N', M_0'; R') and (N", M_0"; R") are live and reversible. For (N, M_0; R), where R_F = { r_2, r_3, r_5, r_6 }, no R_F-siphons would eventually become empty. According to Property 3.4, (N, M_0; R) is also live and reversible. Hence, it may be concluded that the integrated system is live, bounded, reversible and conservative. In other words, the integrated system is well-behaved.



Semantic meaning of places

p_{11}	C_1 is at idle state
p_{12}	C_1 is performing operation o_{11}
p_{13}	C_1 is performing operation o_{12}
p_{14}	S_1 is being withheld
p_{15}	S_2 is being withheld
p_{21}	C_2 is at idle state
p_{22}	C_2 is performing operation o_{21}
p_{23}	C_2 is performing operation o_{22}
p_{24}	S_3 is being withheld
p_{25}	S_4 is being withheld
p_{31}	C_3 is at idle state
p_{32}	C_3 is performing operation o_{31}
p_{41}	C_4 is at idle state
p_{42}	C_4 is performing operation o_{41}
s_1	S_1 is available
s_2	S_2 is available
s_3	S_3 is available
s_4	S_4 is available
s_5	S_5 is available
s_6	S_6 is available

Semantic meaning of transitions

t_{11}	C_1 starts operation o_{11}
t_{12}	C_1 finishes operation o_{11} and starts operation o_{12}
t_{13}	C_1 finishes operation o_{12}
t_{14}	C_1 receives signal m_{11}
t_{15}	C_1 receives signal m_{12}
t_{16}	C_1 receives signal m_{13}
t_{17}	C_1 receives signal m_{14}
t_{21}	C_2 starts operation o_{21}
t_{22}	C_2 finishes operation o_{21} and starts operation o_{22}
t_{23}	C_2 finishes operation o_{22}
t_{24}	C_2 receives signal m_{21}
t_{25}	C_2 receives signal m_{22}
t_{26}	C_2 receives signal m_{23}
t_{27}	C_2 receives signal m_{24}
t_{31}	C_3 starts operation o_{31}
t_{32}	C_3 finishes operation o_{31}
t_{41}	C_4 starts operation o_{41}
t_{42}	C_4 finishes operation o_{41}

Fig. 8. The final composite augmented marked graphs (N, M_0; R)

5 Conclusion

We investigate the property-preserving composition of augmented marked graphs and its application to the synthesis of distributed systems. It is shown that, in composing two augmented marked graphs via their common resource places, boundedness and conservativeness are preserved while liveness and reversibility are preserved under a pretty simple condition. By modelling the distributed system components as augmented marked graphs with common resources denoted by resource places, an integrated system can be obtained by composing these augmented marked graphs via the common resource places. Based on preservation of properties, liveness, boundedness, reversibility and conservativeness of the integrated system can be readily derived.

Liveness, boundedness, reversibility and conservativeness are essential properties that collectively characterise a well-behaved system. For distributed systems which usually involve concurrent (parallel) and asynchronous processes, as competition of common resources exists, it is important for one to assure design correctness in the sense that these essential properties are maintained. By making good use of the special structure and properties of augmented marked graphs as well as the property-preserving composition of augmented marked graphs, our method effectively solves the problem of ensuring design correctness in the composition of distributed system components, which has perplexed designers of distributed systems for a long time.

References

1. Chu, F., Xie, X.: Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming. IEEE Transactions on Robotics and Automation 13(6), 793–804 (1997)
2. Cheung, K.S.: New Characterisations for Live and Reversible Augmented Marked Graphs. Information Processing Letters 92(5), 239–243 (2004)
3. Cheung, K.S., Chow, K.O.: Cycle Inclusion Property of Augmented Marked Graphs. Information Processing Letters 94(6), 271–276 (2005)
4. Cheung, K.S., Chow, K.O.: Analysis of Capacity Overflow for Manufacturing Systems. In: Proceedings of the IEEE Conference on Automation Science and Engineering, pp. 287–292. IEEE Press, Los Alamitos (2006)
5. Cheung, K.S., Chow, K.O.: Compositional Synthesis of Augmented Marked Graphs. In: Proceedings of the IEEE International Conference on Control and Automation, pp. 2810–2814. IEEE Press, Los Alamitos (2007)
6. Huang, H.J., Jiao, L., Cheung, T.Y.: Property-Preserving Composition of Augmented Marked Graphs that Share Common Resources. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, pp. 1446–1451. IEEE Press, Los Alamitos (2003)
7. Reisig, W.: Petri Nets: An Introduction. Springer, Heidelberg (1985)
8. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77(4), 541–580 (1989)
9. Heineman, G.T., Councill, W.T.: Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley, Reading (2002)
10. Crnkovic, I., Larsson, M.: Building Reliable Component-Based Software Systems, Artech House (2002)

A Distributed Scheduling Algorithm in Central-Stage Buffered Multi-stage Switching Fabrics

Yuxiang Hu, Fang Dong, and Julong Lan

National Digital Switching System Engineering & Technological Research Center
Zhenzhou, Henan, P.R. China, 450002
{huyuxiang1982,chxachxa}@yahoo.com.cn

Abstract. The current MSM switching fabric has poor performance under unbalanced traffic. To eliminate the internal congestion of switching fabric, we put forward a new central-stage buffered multi-stage switching fabric—CB-3Clos and the backpressure-based strategy to control flows under credit-dispensed mode. By analyzing the condition to satisfy the central-stage load balance, we also advance an iSLIP alike scheduling algorithm—RGA. The simulation results show: compared with CRRD algorithm based on MSM switching fabric, the RGA algorithm has high throughput irrespective with the arriving traffic model and better performance in packet delay. At the same time, the QoS can be guaranteed.

1 Introduction

Current network faces the embarrassment of unbalance between transport ability and switching ability. Compared with the transport ability developing at a fast speed, the lag of switching ability of core nodes has became the bottleneck of network. The single-stage switches and the relevant scheduling algorithms have became mature, so it's difficult to increase the number of ports or heighten the line-rate to satisfy the requirements of large-scale switching system, the single-stage switch meets it's bottleneck. At present, in order to achieve large-capability and high-expansibility, the development shows an obvious trend: multi-stage switching fabric. Its purpose is taking advantages of multi-stage switching fabrics to design switch systems with large-capability and high-expansibility.

Current researches on switching fabrics mostly focus on Clos network, Benes network, Banyan network and so on. In all of these networks, the three-stage Clos network has become the emphasis owing to its high-modularization and large-capability and the characteristic of strict non-blocking in internal links.

The three-stage symmetrical Clos network C(n, m, r) has r n×m switch cells in the input-stage, and m r×r switch cells in the central-stage, so there are r m×n switch cells in the output-stage. The network has N=n×n input/output ports in all, and every switch cell in central-stage connects with every switch cell both in input-stage and output-stage through a link. It has been proved that [1]: if $m \geq n$, the C(n, m, r) Clos network is a rearrangeable non-blocking switching fabric, this indicates that if and only if there are a pair of matched input/output ports and a pair of unmatched

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 121–130, 2007.

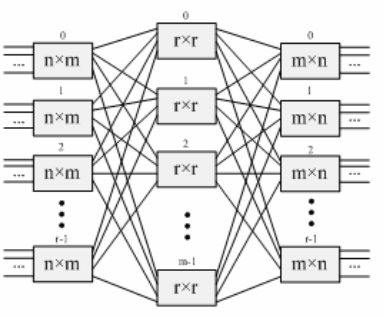


Fig. 1. The three-stage symmetrical Clos network C(n, m, r)

input/output ports, by re-routing between the matched ports, we can establish a match between the pair of unmatched ports. Figure 1 shows the three-stage symmetrical Clos network C(n, m, r).

2 Related Work

2.1 Buffer Setting

The Clos network in pocket switching system falls into two categories: the first is SSS Clos fabric [2]. This fabric takes advantages of space-division multiplexing and makes the bufferless Crossbar as the switch cell. The SSS Clos fabric is simple in physics and has loose requirements on circuit level, but it's strict with the scheduling algorithms and makes them difficult to implement.

With the development of circuit level, buffers have been applied into multi-stage fabric, so the MSM Clos fabric [3] became feasible. The MSM Clos fabric contains buffers in the first and last stage, but use bufferless central-stage. It has been proved that: under uniform traffic, the CRRD algorithm based on MSM can provide nearly 100% throughput, but the performance drops dramatically under unbalanced traffic.

2.2 Flow Control

2.2.1 Backpressure

Backpressure protocol provides a direct method to prevent data from overflow at input ports. It's similar to the counterpressure brought by flows in the pipeline, when the end is closed, the flows will generate counterpressure to the source, and so interdict or slow down the flows. Similarly, the congested ports will send the congestion messages to the sources and make the sources restrict pockets from being sent into network.

Backpressure technique can be selective applied on certain logical links; it's convenient to manage these links between two nodes. The backpressure-based flow control is a good mechanism in the case of whole net, and can be applied in network infrastructures that allow hop-by-hop flows such as the routers, but is restricted in this.

2.2.2 Credit-Based Flow Control

Another universal mechanism is credit-based flow control [5]. Its essence: before send any packet, the sources must receive the credit messages from receivers and the credits decide the number of packets can be sent. Generally, the credit equals to the line-rate multiplying the RTT (round-trip time).

Ideally, the credit-based flow control mechanism can make sure no packet lost, even if under burst traffic, because the length of packet queue will not exceed the credit. Due to this strategy would keep a queue for per link and makes the buffer bigger, it's not suitable for multi-link, and otherwise the buffer will be very complex. The credit-based flow control is usually implemented in low-cost adapters and can achieve high performance.

2.2.3 Regional Explicit Congestion Notification (RECN)

To resolve the internal blocking in multi-stage switching fabric, it's pivotal to resolve the sharing queues between congested and non-congested flows while ensuring the performance. J. Duato and I. Johnson put forward an extensible congestion control strategy [6]. They advise: all non-congested flows share a single queue, while dynamically assign a set-aside-queue for per congestion tree. Congestion trees may be rooted at any output or internal fabric link, and their appearance is signaled upstream via regional explicit congestion notification (RECN) messages. This queue-sharing method based on space-division multiplexing resolves the effects between congested flows, while the complexity of the algorithm is high. The packet delay generated by the transport of the signals is also another problem.

3 Scheduling Algorithm in Multi-stage Switching Fabrics

3.1 Buffer Assignment

The switching fabric may be buffered or bufferless. The bufferless switching fabrics merely steer the flows and couldn't generate any packet delay or output congestion, but it makes the scheduling algorithms very complex and impractical. The buffered switching fabric can resolve the port collision to some extent by setting buffers in the fabric, and current circuit level makes this fabric feasible. Buffer has become an important factor in deciding the performance of the switching fabrics.

Based on the analysis above, inheriting the thinking of space-division multiplexing in multi-stage fabrics, and using the single-stage switches for reference, we put forward a new central-stage buffered three-stage Clos switching fabric—CB-3Clos, figure 1 shows the configuration.

As shown in figure 2, VOQs are set in the input-ports of the switch cells at input-stage to store the congested flows. Here we use the two-stage switching for reference and set buffers in the input-ports of the switch cells at central-stage. So the first and second stage can be treated as a two-stage switching. For the sake of internal non-blocking, the fabric introduces multi-route. Per traffic is allotted to all switch cells of the central-stage uniformly, in a way such as to equalize the rates of the resulting sub-flows. The central-stage switch cells can be thought of as parallel slices of one, faster virtual switch, and inverse multiplexing performs load balancing among these slices.

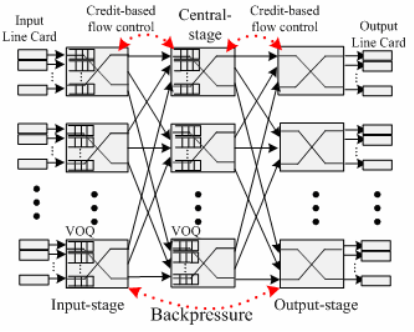


Fig. 2. The configuration of CB-3Clos switching fabric

By setting buffers at the central-stage, the complexity of algorithms on the first stage is reduced efficiently, and it can decrease the unfairness of resources at central-stage brought by the unbalanced dispersed packets' arriving from input-stage. So, to some extent, the load balance on the first two stages can tidy the traffic—tidying the unbalanced traffic into regulated traffic.

3.2 Congestion Control

The scheduler of switch is a buffer-disperser for packets in essence, and the scheduling strategy is just a traffic control strategy selected to guarantee the performance. Considering current traffic control strategies, based on the configuration of CB-3Clos, we put forward a strategy to control traffic which is based on backpressure protocol while in credit-dispensed mode: use backpressure protocol in interior of the whole fabric but in credit-dispersed mode between two conjoint stages. As the broken lines shown in figure 2, firstly the input ports of input-stage send requests to their corresponding ports duo to the queue's estate, secondly the switch cells of output-stage produce credits by calculating the queues and the buffers, and then send credits to central-stage, after receive the credits, the switch cells of central-stage send them to input-stage by load balance. Only the input ports that have obtained credits can send packets into switching fabric.

Traffic first being store in buffers then entering fabric, this can obtain high throughput and small buffer in interior of the fabric. It's a typical thinking of space-division multiplexing, and it makes only the traffic destined out can occupy the buffer, while the congested traffic stay out of fabric. At the same time, to make full use of the buffer which is very costly, the buffer is reserved from the output stage to input stage, once a stage. This is precisely opposite to how packets progress under backpressure protocol. The direction chosen ensures that each reservation, when performed, is on behalf of a packet that is guaranteed not to block inside the buffer: buffer space has already been reserved for that packet in the next downstream buffer. Hence, packets will be allowed to move freely, and without danger of any buffer overflowing. Figure 3 gives a demonstration of the congestion control.

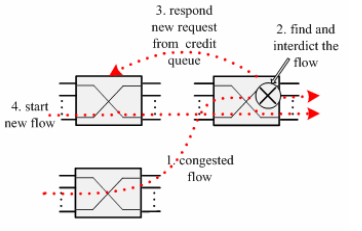


Fig. 3. Demonstration of the congestion control in interior of the fabric

This strategy may lead to the phenomenon that several ports reserve buffers for the same packet; here we define this phenomenon as output buffer collision. The output buffer collision may lead to: buffer reservations constitute a second pass through the fabric, after requests have traversed once from inputs to the per-output scheduler. So the most essential method is making sure that no other input ports send any request to these output ports.

In sum, we give the rules to resolve the congestion between flows: in per input/output port, per credit scheduler work paralleling in pipe-line independently; the request must fix on certain output port for per flow and be sent to corresponding port; requests must queue in the credit scheduler, per output scheduler responds to these requests after assigns buffers. Credits may be generated based on certain QoS strategies such as: WRR/ WFQ. When packet comes out of the fabric, it must inform the credit scheduler to re-assign buffer, so the department rate can regulate the rate that switch be granted.

3.3 Load Balance in Central-Stage

In multi-path fabrics, route can be selected by credit scheduler. To guarantee internal non-blocking in switching fabric, per flow should be allotted to all switch cells in central-stages averagely.

Suppose a certain output port and an ideal traffic model, we follow the filter rule when allotting the traffic from certain input port to all switch cells in central-stage. And we also suppose: in a slot, there are k ports in central-stage is free, R is the bandwidth of the whole fabric, ΔB is the bandwidth allocation parameter of certain flow, A_x stands for the x th switch cell in input-stage, while B_y stands for the y th switch cell in central-stage, the credit scheduler in input-stage calculate the weighted credit that it received. If in a slot, the result satisfies the rule I:

$$l(A_x \to B_y, T) = \max\{w(A_x B_y)\} \tag{I}$$

At the same time, the result satisfies the rule II:

$$\sum l(A_x \to B_y, T) \le \frac{R}{k} + \Delta B \tag{II}$$

Then the scheduler will acknowledge the corresponding grant.

In other words, the allotment of grants and load must make sure that all flows on $A_x \to B_y$ are restricted in $R/k + \Delta B$ bandwidth. The grants in central-stage are generated independently. Easy to say, if only there are free buffers, the link between $A_x \to B_y$ will never be free. So in this ideal traffic model, all flows to enter the fabric will find free buffer. Figure 4 gives a simple example.

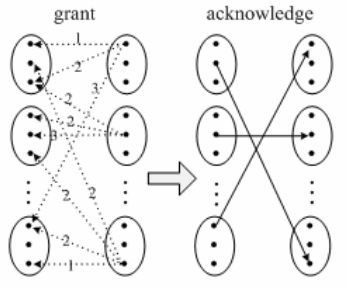


Fig. 4. A simple example of load balance

In ideal stream model, it's easy to achieve perfect load balance by rule I, but in practical system, several inputs of the same switch cell may send concurrently to the same switch cell, which is inevitable under distributed and independent load-balancing. So in order to deal with quantization imbalance, all flows between the $A_x \to B_y$ must satisfy rule II, and as a result, buffers must be set at the input-stage to store the temporary congested flows.

There are many advantages taking algorithm for load-balance in central-stage: the algorithm is brief, expansible and prone to be implemented by hardware; the algorithm guarantees the fairness by setting the upper limit of bandwidth, while the credits are generated based on certain QoS strategy, so it can efficiently support quality of service also; the fabric achieves perfect load-balancing for per packet and can control the congested flow in real-time.

3.4 Analysis of the Scheduling Algorithm

Based on the analysis above, we put forward an iSLIP alike scheduling algorithm based on the CB-3Clos switching fabric—RGA algorithm. It takes advantages of parallel iterative matching algorithm—PIM and round-robin matching algorithm—iSLIP, adopts the PIM's "request-grant-accept" mechanism and iSLIP's "Round-Robin" mechanism, is distributed implemented in credit schedulers, so this algorithm can achieve fairness and stabilization while keeps high efficiency.

The RGA algorithm still has three steps in sum: they are request, grant and acknowledgement.

Step 1: request. All input ports whose VOQ is busy send requests to the corresponding output ports.
Step 2: grant. Once receive several requests from input ports, the credit schedulers in output-stage generate credits based on the state of buffers and queues under certain QoS strategy. After inform the credits to the input ports, the credit schedulers increase the poll point by one (mol N).
Step 3: acknowledgement. After receive the grants, the unmatched input ports select the output ports satisfying the rule I and II from the credit queues to acknowledge. If it's the first time, the input ports and the acknowledged output ports should modify the poll point. The modification rule is: the points of both input and output port increase by one (mol N). After each match, all unmatched ports turn into the next match.

4 Analysis of Simulation

To illuminate the performance of RGA algorithm, we establish the simulation model. Under this model we get the throughout and packet delay and then compare them with the performance of CRRD algorithm in MSM fabric, All through the simulation supposing $m = 2n$.

4.1 Traffic Model

First we define the traffic model in the simulation model.

- Uniform traffic: the ports aimed by arriving flows distribute uniformly, and all ports face the same traffic load.
- Unbalanced traffic: the load ρ between input port s and output port d follows:

$$\rho_{s,d} = \begin{cases} \rho\left(\omega + \dfrac{1-\omega}{N}\right), & if \quad s = d \\ \rho\dfrac{1-\omega}{N}, & otherwise \end{cases} \tag{1}$$

ω stands for the unbalanced factor, N stands for the number of ports, so the load on d is:

$$\rho_d = \sum_s \rho_{s,d} = \rho\left(\omega + N\frac{1-\omega}{N}\right) = \rho \tag{2}$$

In the model, we take the Bernoulli traffic and burst traffic, In the same period of ON, the destination addresses are same, while in different period, the destination addresses are distributed in uniform and unbalanced model, the length of both ON and OFF state follow the geometry distribution, by the traffic load ρ and average burst length E_{om}, we will get:

$$\rho_d = \frac{1}{E_{ON}}, q = \frac{\rho \times p}{1 - \rho + \rho \times p} \tag{3}$$

4.2 Performance Comparison Between Algorithms with Fix-Size Packets Under Different Traffic Models

4.2.1 Uniform Bernoulli Traffic Model

Table 1 shows the throughout of RGA and CRRD algorithm under fabrics of different scales by uniform Bernoulli traffic model, it's easy to say that both algorithms achieve high throughout, especially RGA algorithm nearly get 100% throughout.

Table 1. Throughout of RGA and CRRD algorithm under fabrics of different scales by uniform Bernoulli traffic model

	64×64	128×128	256×256
CRRD	96.4%	97.1%	95.2%
PGA	99.3%	99.1%	98.9%

4.2.2 Unbalanced Traffic Model

Under unbalanced traffic model, the performance of RGA and CRRD algorithm shift dramatically.

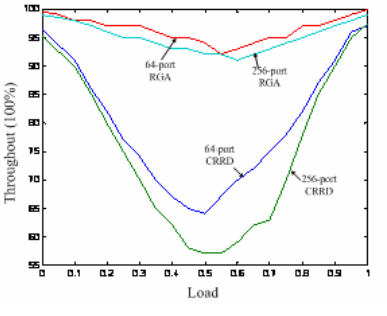


Fig. 5. Throughout of algorithms under fabrics of different scales by unbalanced traffic model

As figure 5 shows, under fabrics of different scales by unbalanced traffic model, throughout of both algorithms drop fast, and achieve the minimum 57% when ω is 0.5 and then ascend. But noticeably, the throughout of RGA algorithm holds above 90% all the time, which illuminates that the throughout of RGA algorithm is irrespective with arriving traffic model.

Figure 6 gives the packet delay of algorithms under fabrics of different scales by unbalanced traffic model. It's obvious that the packet delay increases with ω fast.

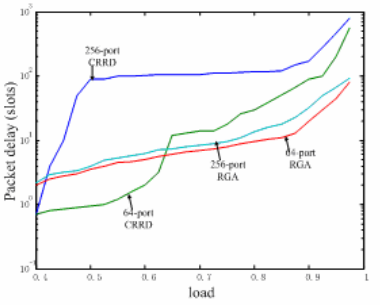


Fig. 6. Delay of algorithms under fabrics of different scales by unbalanced traffic model

Under 64-ports, when ω is under 0.65, the packet delay of RGA is a little bigger than CRRD's, but when ω exceed 0.65, the packet delay of RGA is smaller than CRRD's a lot. It's the same under 256-ports. This phenomenon can be explained by the buffers set in the central stage of switching fabric. To achieve better load balance, the buffers will work out certain delay, but while ω exceeds certain ambit, the impact of buffers reduce.

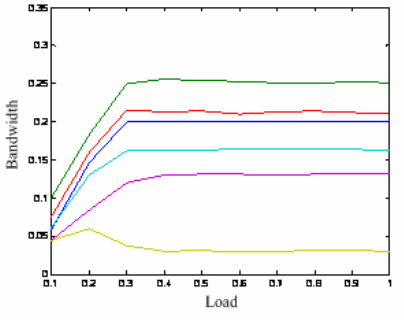


Fig. 7. The bandwidth allocation of RGA

Figure 7 gives the bandwidth allocation of RGA. Supposing the arriving rate of EF and AF is: 18%, 24%, 20%, 16%, 12%, 10%. The corresponding bandwidth allocation is: 19.8%, 24%, 20%, 16%, 12%, 8.2%. From the figure 7 we can see: the performance on bandwidth allocation of RGA is good.

5 Conclusions and Future Work

On the basis of analysis of congestion control theory, we put forward a new central-stage buffered three-stage Clos switching fabric—CB-3Clos and the backpressure-based flow control strategy under credit-dispensed mode. By analyzing the condition to satisfy the central-stage load balance, we also advance an iSLIP alike scheduling algorithm—RGA. The simulation results show: compared with CRRD algorithm of

MSM Clos fabric, the RGA algorithm has high throughput irrespective with the arriving traffic model and better performance in packet delay. At the same time, the QoS can be guaranteed.

By setting buffers at the central-stage, we can reduce the complex of algorithm while improve the performance of switching fabrics, and current circuit level make it possible. But this fabric with multi-route may lead packets to out-of-order, how to keep the order of packets in multi-stage fabrics will be our emphases of researches in the future.

Acknowledgment

This paper is jointly funded by Chinese National High Technology Research and Development Program (NO. 2005AA121210) and the National Science Foundation of China (NO. 60572042)

References

1. Clos, C.: A study of nonblocking switching fabric networks [J]. BSTJ 32(5), 406–424 (1953)
2. Chao, H.J., Deng, K., Jing., Z., Petabit, A.: Photonic Packet Switch (P3S) [C]. In: Proc. IEEE Infocom 2003, vol. 21(7), pp. 1096–1112. IEEE Computer Society Press, Los Alamitos (2003)
3. Oki, E., Jing, Z., Rojas-Cessa, R., et al.: Concurrent Round-Robin-Based Dispatching Schemes for Clos-Network Switches [J]. IEEE/ACM Trans. on Networking 10(2), 830–844 (2002)
4. Sapountzis, G., Katevenis, M.: Benes Switching Fabrics with O(N)-Complexity Internal Backpressure [J]. IEEE Communications Magazine 43(1), 88–94 (2005)
5. Kong, H.T., Morris, R.: Credit-Based Flow Control for ATM [J]. IEEE Magazine 9(2), 40–48 (1995)
6. Duato, J., Johnson, I., Flich, J., et al.: A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks [C]. In: Proc. HPCA-11, San Francisco, USA, pp. 108–119 (February 2005)
7. Chrysos, N., Katevenis, M.: Scheduling in Switches with Small Internal Buffers [C]. In: Proc. IEEE Globecom2005, MO, USA, pp. 614–619 (2005)
8. Chang, C., Chen, W., Juang, H.: On Service Guarantees for Input Buffered Crossbar Switches: A Capacity Decomposition Approach by Birkhoff and von Neumann [C]. In: Proceedings of IEEE IWQoS, pp. 79–86 (1999)
9. Chao, H.J., Park, J.S.: Centralized contention resolution schemes for a large-capacity optical ATM switch [C]. In: Proc. IEEE ATM workshop, pp. 11–16 (1998)

Improving Recovery in Weak-Voting Data Replication*

Luis H. García-Muñoz, Rubén de Juan-Marín, J. Enrique Armendáriz-Íñigo, and Francesc D. Muñoz-Escoí

Instituto Tecnológico de Informática - Universidad Politécnica de Valencia
Camino de Vera, s/n - 46022 Valencia, Spain
{lgarcia,rjuan,armendariz,fmunyoz}@iti.upv.es

Abstract. Nowadays eager update everywhere replication protocols are widely proposed for replicated databases. They work together with recovery protocols in order to provide highly available and fault-tolerant information systems. This paper proposes two enhancements for reducing the recovery times, minimizing the recovery information to transfer. The idea is to consider on one hand a more realistic failure model scenario –crash recovery with partial amnesia– and on the other hand to apply a compacting technique. Moreover, it is provided amnesia support avoiding possible state inconsistencies –associated to the failure model assumed– before starting the recovery process at recovering replicas.

1 Introduction

Database replication consists in maintaining identical copies of a given database at multiple network nodes. This improves performance, since clients access their local replica or are forwarded to the less loaded one; and availability: whenever a node fails, its associated clients are silently redirected to another available one. Replication protocols can be designed for eager or lazy replication [1], and for executing updates in a primary copy or at all node replicas [2]. With eager replication we can keep all replicas exactly synchronized at all nodes, but this could have an expensive cost. With the lazy alternative we can introduce replication without severely affecting performance, but it can compromise consistency. Many replication protocols are based on eager update everywhere with a *read one, write all available* (ROWAA) approach. As we have briefly highlighted before, these replication protocols provide high availability. However, only a few of them deal with the possible reconnection of the failed node, which is managed by recovery protocols [3,4,5,6].

The aim of the recovery protocols is to bring failed or temporarily disconnected nodes back into the network as fully functional peers, by reconciling the database state of these recovering nodes with that of the active nodes. This could be done by logging transactions and transferring this log to recovering nodes so they can process missed transactions, or transferring the current state of the items that have been updated in the database since the recovering node failed.

* Work supported by FEDER, the Spanish MEC grant TIN2006-14738-C02 and the Mexican DGEST and SES-ANUIES.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 131–140, 2007.

This paper is focused in the recovery protocol for eager update everywhere replication protocols, proposing some optimizations to the work presented in [6]. These enhancements include amnesia support, and a better performance reducing the amount of data to save in the actions done before recovering and the amount of data to transfer at recovering time. The main idea in the last case is to compact recovery data eliminating redundant information.

The rest of this paper is distributed as follows. Section 2 provides the system model. Section 3 deals with the basic recovery protocol. Section 4 explains the necessary actions for the amnesia support. Next, Section 5 relates the process of compacting recovery information. Later, Section 6 shows the simulation results followed by the related works in Section 7. In the final Section 8, we provide our conclusions.

2 System Model

The basic recovery protocol has been designed for database replicated systems composed by several replicas –each one in a different node–. These nodes belong to a partially synchronous distributed system: their clocks are not synchronized but the message transmission time is bounded. The database state is fully replicated in each node.

This replicated system uses a group communication system (*GCS*) [7]. Point-to-point and broadcast deliveries are supported. The minimum guarantee provided is a FIFO and reliable communication. A group membership service is also assumed, that *knows* in advance the identity of all potential system nodes. These nodes can join the group and leave it, raising a *view change event*. Therefore, each time a membership change happens, i.e. any time the failure or the recovery of one of the member nodes occurs, it supplies consistent information about the current set of reachable members as a view. The group membership service combined with the *GCS* provides *Virtual Synchrony* [7] guarantees, which is achieved using *sending view delivery* multicast [7] enforcing that messages are delivered in the view they were sent. A *primary component* [7] model is followed in case of network partitioning.

The replicated system assumes the *crash-recovery with partial-amnesia* [8] model. This implies that an outdated node must be recovered from two "lost of updateness": forgotten state and missed state. This assumption supports a more realistic and precise way to perform the recovery process. So the assumed model allows to recover failed nodes from their previous crashing state maintaining their assigned node identifiers.

3 Basic Recovery Protocol

Our basic proposal is inspired in the recovery protocol presented in [6]. It has been designed for *eager update everywhere* database replication protocols and proposes the use of *DB-partitions* (see below). It was originally designed for providing recovery support for the *ERP* and *TORPE* [6] replication protocols. Such protocols use a voting termination approach [2], and can be considered as weak voting replication protocols [9]. This basic recovery protocol can be outlined as follows:

– The system has a database table named $MISSED$, which maintains all the information that will be needed for recovery purposes. Each time a new view is installed

a new entry is inserted in the $MISSED$ table if there are failed nodes. Each entry in $MISSED$ table contains: the view identifier, the identifiers of crashed nodes in this view –$SITES$–, and the identifiers list of data items modified during this view –OID_LIST–. The two first ones are set at the beginning of the view, while the last one grows as long as the view passes.

- When a set of crashed nodes reconnects to the replicated system, the recovery protocol will choose one node as the *recoverer* with a deterministic function. Then in a first step the *recoverer* transfers the metadata recovery information to all reconnected nodes. This metadata information contains: the identifiers of modified items, and the crashed node identifiers in each view lost by the oldest crashed node being recovered. The per-view metadata generates a DB-partition during the recovery process; i.e., such items will be blocked while they are being transferred to the recovering node, logically partitioning the database. These *DB-partitions* are also used in order to block in each replica the current user transactions whose modified items conflict with its *DB-partitions*. Subsequently, the *recoverer* starts to recover each *recovering* node view by view. For each lost view, the *recoverer* transfers the state of the modified items during this view. And, once the view has been recovered in the *recovering* node, it notifies the recovery of this view to all alive nodes. The recovery process ends in each *recovering* node once it has updated all its lost views.
- As a transaction broadcast is performed spreading two messages –*remote* and *commit*–, it is possible that a reconnected node receives only the second one, without any information about the updates to be committed. In this case the replication protocol will transfer the associated writesets to these nodes. This behavior implies that transaction writesets are maintained in the sender node until the *commit* message is broadcast.

But this recovery protocol presents the following two problems:

- Amnesia phenomenon. Although we are assuming the *crash-recovery with partial amnesia* [8] failure model, many systems do not handle it in a perfect way. This problem arises because once the replication protocol propagates the *commit* message associated to one transaction, and it is delivered, the system assumes that this transaction is being committed locally in all replicas. But this assumption even using strong virtual synchrony [7] is not always true. It is possible that a replica receives a transaction *commit* message, but before applying the commit the replica crashes, as it is commented in [10] –the basic idea is that message delivery does not imply correct message processing–. The problem will arise when this crashed node reconnects to the replicated system, because it will not have committed this transaction and the rest of the system will not include among the necessary recovery information the updates performed by this transaction, arising then a problem of replicated state inconsistency.
- Large $MISSED$ table and redundant recovery information. If in the system there are long-term crashed nodes –meaning nodes failed during many views– and there are also high update rates it is possible that the $MISSED$ table enlarges significantly with high levels of redundant information, situation that is strongly discouraged. Redundant recovery information will appear because it is possible that the

same item has been modified in several views where the crashed nodes set is very similar. In this case if an item is modified during several views, only knowing the last time –meaning the last view– it was updated is enough. Therefore, it will be interesting to apply algorithms that avoid redundant recovery information, because the larger $MISSED$ tables the greater the recovery information management overhead becomes.

In the following section we will present and study different approaches for solving these problems improving the basic recovery protocol.

4 Amnesia Support

In order to provide amnesia support different approaches can be considered. These approaches can be classified depending on which recovery information they use. On one hand, there are the ones using the broadcast messages –log-based– [3,4] and, on the other hand there are the ones using the information maintained in the database –version-based– [5,6].

But before describing how the amnesia support can be provided in the basic recovery protocol, it must be considered how this amnesia phenomenon manifests. In [11], it is said that the amnesia phenomenon manifests at two different levels:

- *Transport level*. At this level, amnesia implies that the system does not remember *which messages have been received*. In fact, the amnesia implies that received messages non-persistently stored are lost when the node crashes, generating a problem when they belong to transactions that the replicated system has committed but which have not been already committed in the crashed node.
- *Replica level*. The amnesia is manifested here in the fact that the node "forgets" *which were the really committed transactions*.

Hereafter we detail a log-based solution for the amnesia problem. There are other amnesia supporting techniques –e.g., a version-based approach [5]– but are not presented here to due space constraints.

The information maintained in order to perform the amnesia recovery process will be the broadcast replication messages. In this replication protocol two messages for each propagated transaction: *remote* and *commit*. The amnesia recovery must be performed before starting the recovery of missed updates –the latter will be done by the basic recovery protocol–. The amnesia recovery process will consist in reapplying the messages belonging to non really committed transactions.

A transport-level solution consists in each node storing persistently the received messages, maintaining them as long as the associated transaction, t, has not been committed and discarding them as soon as t its really committed in the replica. But, the message persist process must be performed atomically inside the delivery process as already discussed in [10] with its "successful delivery" concept. Moreover, messages belonging to aborted or rolled-back transactions must be also deleted.

Once the amnesia phenomenon is solved at transport level, it is necessary to manage the amnesia problem at replica level. At this level the amnesia implies that the system

can not remember which were the really committed transactions. Even for those transactions for which the "commit" message was applied, it is possible for the system to fail *during* the commit. Then the amnesia recovery process in a replica will consist in reapplying (and immediatly deleting, in the same transactional context) the received and persistently stored messages in this replica that have not been already deleted, because it implies that the corresponding transactions have not been committed in the replica. These messages are applied in the same order as they were originally received.

It also must be noticed, that in this process is not needed to apply the *remote* messages whose associated *commit* messages have not been received, because it implies that they have been committed in the subsequent view, and therefore their changes are applied during the recovery of its first missed view.

Finally, once the amnesia recovery process ends, the basic recovery protocol mechanism can start.

5 Compacting Recovery Information

In order to increase the performance at the moment of determining and transferring the necessary information for the synchronization of recovering nodes, we propose some modifications based on packing information that enhance the basic recovery protocol described in [6]. This could be done by compacting the records in the $MISSED$ table, and with this, minimize the items to transmit and to apply them in the recovering node, reducing thus the transmission and synchronization time.

These item identifiers can be packed due to the fact that the recovery information only maintains the identifiers of updated items. The state of these items is retrieved by the *recoverer* from the database at recovering time. Moreover, if a *recovering* node, k, has to recover the state of an item modified in different views lost by k it will receive as many times the item value, but transferring its state only once is enough. As a consequence, it is not relevant to repeat the identifier of an updated item across several views, being only necessary to maintain it in the last view it was modified.

We consider that the actions for the amnesia support are performed during the execution of user transactions. Whenever one (or more than one) node fails, the recovery protocol starts the execution of the actions to advance the recovery of failed nodes. To this end, when a transaction commits, the field which contains the identifiers of the updated items, OID_LIST, will be updated in the following way:

1. For each item in the $WriteSet$, the OID_LIST is scanned to verify if the item is already included in it or not. If it is not, it is included and is looked for in previous views OID_LIST, eliminating it from the OID_LIST in which it appears, compacting thus the OID_LIST, i.e. the information to transfer when a node recovers.
2. If as a result of this elimination, an OID_LIST is emptied, the content of the field $SITES$ is included into the field $SITES$ of the next record, and the empty record in the table $MISSED$ can be eliminated.

When a node reconnects to a replicated system, the new view is installed and the actions for the amnesia recovery are performed locally at the recovering node. This is

a lightweight process (i.e. only a few stored messages have to be processed) in comparison to the database state recovery process itself. The other nodes know who is the recovering node, and every one performs locally the next actions:

1. The $MISSED$ table is scanned looking for the recovering node in the field $SITES$ until the view that contains the recovering node is found. The items for which the recovering node needs to update its state are the elements of OID_LIST of this view and the subsequent views.
2. At the recoverer node, the recovery information is sent to the recovering node according to the basic protocol.
3. Once the recovering node has confirmed the update of a view, the node is eliminated from the $SITES$ field in this view, and if it is the last item, also the record that contains this view is eliminated.
4. If a recoverer node fails during the recovering process, then another node is elected to be the new recoverer, according to the basic protocol. And it will create the partitions pending to be transferred, according to the previous points, and then it will perform the item transfer to recovering nodes, again as in the basic protocol.

It is important to note that in a view change consisting in the join and leave of several nodes, we must first update the information about failed nodes, and later execute the recovery process.

6 Simulation Results

We have simulated the compacting enhancement in order to know which level of improvement provides. We have considered three replicated scenarios with 5, 9 and 25 nodes each one. The replicated database has 100000 data items. All simulations start having all replicas updated and alive. Then, we start to crash nodes one by one –installing a new view each time a node crashes–, until the system reaches the minimum primary partition in each scenario. At this point two different recovery sequences are simulated. In the first one, denoted as order 1, the crashed nodes are reconnected one by one in the same order as they crashed, while in the second, denoted as order 2, they are reconnected one by one but reversing their crash order. In both cases, each time a node reconnects a new view is installed, and immediately the system starts its recovery, ending its recovery process before reconnecting the following one. In any installed view we assume that the replicated system performs 250 transactions successfully, and each transaction modifies 20 database items. All simulation parameters are described in Table 1.

The items in the writeset are obtained randomly with a uniform distribution. We have not used neither a hot spot, as in other previous works [12], nor typical workloads as TPC-W or TPC-C [13]. In both cases, they would be more favorable environments for the compacting method than a uniform distribution, since they suppose more frequent access to a set of items of the database, removing a big amount of items in the compacting proccess. We have also assumed a fast network, and this reduces the performance difference between the normal and compacting recoveries, since it only depends on the

Table 1. Simulator Parameters

Parameter	*Value*	*Parameter*	*Value*
Number of items in the database	100000	Time for a read	4 ms
Number of servers	5, 9, 25	Time for a write	6 ms
Transactions per view	250	Time for an identifier read	1 ms
Transaction length	20 modified items	Time for an identifier write	3 ms
Identifier size	4 bytes	CPU time for an I/O operation	0,4 ms
Item size	200 bytes	Time for point to point message	0,07 ms
Maximum message size	64 Kbytes	Time for broadcast message	0,21 ms
CPU time for network operation	0,07 ms		

Table 2. Recovery times in seconds (N = Nodes, V = Views)

Order	*N*	*V*	*Basic*		*Compacted*		*Order*	*N*	*V*	*Basic*		*Compacted*	
			Avg	*StdDev*	*Avg*	*StdDev*				*Avg*	*StdDev*	*Avg*	*StdDev*
1	5	2	165.8	0.23	161.7	0.21	2	25	5	414.6	0.18	376.1	0.15
2	5	1	82.8	0.20	82.9	0.18	2	25	7	580.4	0.19	502.1	0.14
2	5	3	248.7	0.18	236.7	0.16	2	25	9	746.2	0.19	616.0	0.12
1	9	4	331.6	0.19	308.1	0.18	2	25	11	912.0	0.19	719.2	0.11
2	9	1	82.9	0.17	82.9	0.20	2	25	13	1077.9	0.19	812.6	0.11
2	9	3	248.7	0.17	236.7	0.18	2	25	15	1243.8	0.19	897.1	0.10
2	9	5	414.5	0.18	376.0	0.17	2	25	17	1409.6	0.19	973.6	0.10
2	9	7	580.4	0.18	501.9	0.17	2	25	19	1575.5	0.19	1042.9	0.09
1	25	12	995.1	0.18	767.2	0.12	2	25	21	1741.3	0.19	1105.4	0.08
2	25	1	82.9	0.20	82.9	0.19	2	25	23	1907.2	0.18	1162.0	0.07
2	25	3	248.7	0.19	236.7	0.16							

amount of transferred items. If we had a slow network, such difference would have been bigger. We have made one hundred repetitions for every experiment obtaining with this, the guarantees of a low dispersion (see Table 2).

This simulation has not considered the costs of: managing the amnesia problem, and recovery information compacting. The amnesia problem, as it has been said before, is solved using a log-based approach, persisting the delivered messages during the replication work, and applying those not committed during the amnesia recovery process. Thus, it implies two costs: one in the replication work and another in the recovery work. The first cost is not considered because does not happen in the recovery process. The second one, although appears in the recovery process, is not considered because it is very low compared to the recovery process itself –usually it will consist in applying few messages (writesets) and in our simulation are very small–. The recovery information compacting cost is not taken into account because this work is performed online, therefore its associated overhead penalizes only the replication work performance, but not the recovery.

The simulation results show that the more views a crashed node loses the better the compacting technique behaves, which is a logical result. In fact, when more updates a crashed node misses the probability of modifying the same item increases. Both in the Table 2 and in the Figure 1 we can observe the same behavior. When a crashed node has lost only one view the compacting technique does not provide any improvement because it has been unable to work. But, as long as the crashed node misses more views the compacting technique provides better results.

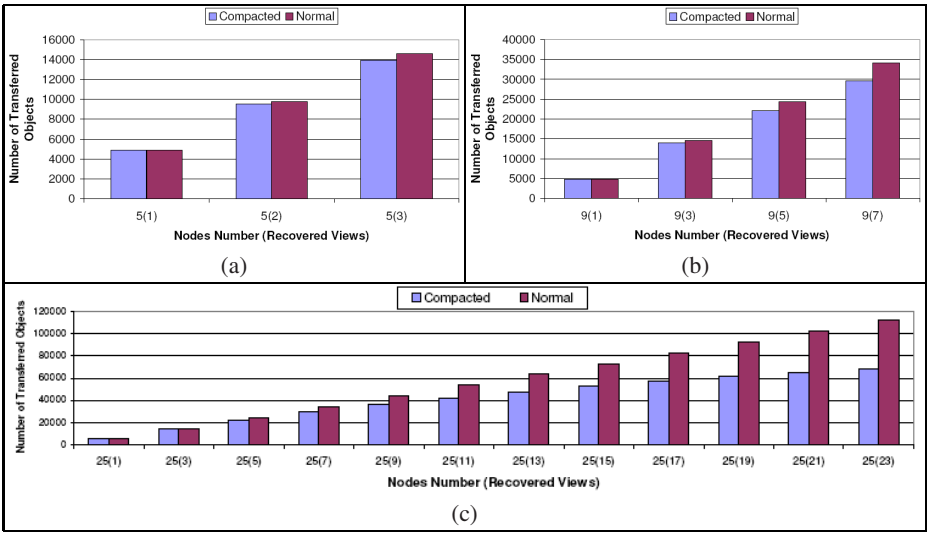


Fig. 1. Item Compactness: (a) 5 nodes, (b) 9 nodes, (c) 25 nodes

It must be also noticed that the basic recovery protocol could arrive to transfer a greater number of items than items has the original database. This occurs because it transfers for each lost view all the modified (and created items in this view) independently they are transferred when recovering other views where these items have been also modified. This situation is avoided by our recovery protocol enhancement. And in the worst case the proposed solution will transfer the whole database because during the inactivity period of the recovered node all the items of the database have been modified.

Obviously, we must say that the improvement provided by our approach depends on the replicated system load activity, the update work rate, and the changed items rate. For the first two ones, we can consider in a general way that when higher they are better our compacting technique behaves. This is because the probabilities of modifying the same item in different views increase. This consideration drives us to the changed items rate, which is really the most important parameter. It tells us if the performed updates are focused in few items or not. Then for our technique it is interesting that changes are focused in as few items as possible. In fact, the worst scenario for our technique will be the one in which all the modifications are performed in different items.

As final conclusion, we can say that our enhanced recovery protocol works better in some of the worst scenarios from a recovery point of view: when the crashed node has lost a lot of updates and the changed items rate is not very high.

7 Related Work

For solving the recovery problem [14] database replication literature has largely recommended the crash recovery failure model use as it is proposed in [3,4,5,6], while process replication has traditionally adopted the fail stop failure model. The use of different approaches for these two areas is due to the fact that usually the first one manages large data amounts, and it adopts the crash recovery with partial amnesia failure model in order to minimize the recovery information to transfer.

The crash-recovery with partial amnesia failure model adoption implies that the associated recovery protocols have to solve the amnesia problem. This problem has been considered in different papers as [10,11,15] and different recovery protocols have presented ways for dealing with it. The *CLOB* recovery protocol presented in [3] and the *Checking Version Numbers* proposed in [5] support amnesia managing it in a log-based and version-based way, respectively.

In regard to the compactness technique, [16] uses it in order to optimize the database recovery. In this case, this technique is used to minimize the information size that must be maintained and subsequently transferred in order to perform the recovery processes. Such paper also presents experimental results about the benefits introduced by using this technique, reaching up to 32% time cost reductions.

The background idea of our compacting technique is very similar to the one used in one of the recovery protocols presented in [5]. This protocol maintained in a database table the identifiers of the modified objects when there were failed nodes. Each one of these object identifiers was inserted in a different row, storing at the same time the identifier of the transaction which modified the object. Therefore, when an object was modified the system checked if its identifier was already inserted in this table. If it has not, the protocol created a new entry where inserted the identifier object and the transaction identifier. If it already existed an entry with this object identifier, the protocol simply updated in this entry the transaction identifier. So, this recovery protocol also avoids redundant information, but it uses a more refined metadata granularity –transaction identifier– than our enhanced protocol –view identifier–.

8 Conclusions

In this paper we have reviewed the functionality of the original recovery protocol described in [6]. We have enhanced it providing an accurated amnesia support and incorporating a compacting method for improving its performance.

The amnesia support has been improved using a log-based technique which consists in persisting the messages as soon as they are delivered in each node, in fact they must be persisted atomically in the delivery process.

Our compacting technique avoids that any data object identifier appears more than once in the *MISSED* table. Then, this mechanism reduces the size of recovery messages, both the ones that set up the DB-partitions and the ones which transfer the missed values.

Tests have been made with a simulation model and the advantages of the enhanced recovery protocol have been verified when comparing the results of both protocols. The obtained results have pointed out how our proposed compacting technique provides better results when the number of lost views by a crashed node increases. Thus, our compacting technique has improved the recovery protocol performance for recoveries of long-term failure periods.

References

1. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: ACM SIGMOD International Conference on Management of Data, pp. 173–182. ACM Press, New York (1996)
2. Wiesmann, M., Schiper, A., Pedone, F., Kemme, B., Alonso, G.: Database replication techniques: A three parameter classification. In: SRDS, pp. 206–215 (2000)
3. Castro, F., Esparza, J., Ruiz, M., Irún, L., Decker, H., Muñoz, F.: CLOB: Communication support for efficient replicated database recovery. In: 13th Euromicro PDP, Lugano, Sw, pp. 314–321. IEEE Computer Society Press, Los Alamitos (2005)
4. Jiménez-Peris, R., Patiño-Martínez, M., Alonso, G.: Non-intrusive, parallel recovery of replicated data. In: SRDS, pp. 150–159. IEEE Computer Society Press, Los Alamitos (2002)
5. Kemme, B., Bartoli, A., Babaoğlu, O.: Online reconfiguration in replicated databases based on group communication. In: Intl.Conf.on Dependable Systems and Networks, Washington, DC, USA, pp. 117–130 (2001)
6. Armendáriz, J.E., Muñoz, F.D., Decker, H., Juárez, J.R., de Mendívil, J.R.G.: A protocol for reconciling recovery and high-availability in replicated databases. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 634–644. Springer, Heidelberg (2006)
7. Chockler, G.V., Keidar, I., Vitenberg, R.: Group communication specifications: A comprehensive study. ACM Computing Surveys 4(33), 1–43 (2001)
8. Cristian, F.: Understanding fault-tolerant distributed systems. Communications of the ACM 34(2), 56–78 (1991)
9. Wiesmann, M., Schiper, A.: Comparison of database replication techniques based on total order broadcast. IEEE Trans. Knowl. Data Eng. 17(4), 551–566 (2005)
10. Wiesmann, M., Schiper, A.: Beyond 1-Safety and 2-Safety for replicated databases: Group-Safety. In: Proceedings of the 9th International Conference on Extending Database Technology (EDBT2004), Heraklion - Crete - Greece (2004)
11. de Juan-Marín, R., Irún-Briz, L., Muñoz-Escoí, F.D.: Supporting amnesia in log-based recovery protocols. In: ACM Euro-American Conference on Telematics and Information Systems, Faro, Portugal, ACM Press, New York (May 2007)
12. Kemme, B.: Database Replication for Clusters of Workstations. PhD thesis, Swiss Federal Inst. of Technology, Zurich, Switzerland (2000)
13. The transaction processing performance council, http://www.tpc.org
14. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison Wesley, Reading, MA, EE.UU (1987)
15. de Juan-Marín, R., Irún-Briz, L., Muñoz-Escoí, F.D.: Recovery strategies for linear replication. In: ISPA, pp. 710–723 (2006)
16. Civera, J.P., Ruiz-Fuertes, M.I.: García-Muñoz, L.H., Muñoz-Escoí, F.D.: Optimizing certification-based database recovery. Technical report, ITI-ITE-07/04, Instituto Tecnológico de Informática (2007)

Exploring Data Reusing of Failed Transaction

Shaogang Wang, Dan Wu, Xiaodong Yang, and Zhengbin Pang

School of Computer, National University of Defense Technology
Changsha, Hunan, 410073 China
wshaogang@nudt.edu.cn

Abstract. Transactional Memory (TM) has been the promising parallel programming technique to relieve the tedious work of synchronizing shared object using lock mechanism. Transaction execution required to be atomic and isolated relative to the whole system. The transaction fails if found violated access to the shared object from other transaction, and it will be re-executed till finally commit successfully; currently, most TM systems are required to restore shared memory's state before re-execution, this cleanup cost and the shared object's opening cost greatly hurdle system's performance.

In this paper, we propose a new general transaction iteration's data reusing (TItDR) method which reuses the opened object of failed transaction in the following re-execution. The obvious advantage is that it greatly simplify the opening process if it has been opened in previous failed transaction and most of the cleanup work are no longer needed. TItDR leaves opened object in pseudo-active state and restart the transaction, We talk about conflicts resolution, validation, commit/abort processing problem along with our data reusing method and show that TItDR will not incur more conflicts and more overhead for validation or commit. Both currently proposed software transactional memory (STM) systems and hardware systems (HTM) have much potential data reusing.

Our test result is based on STM implementation, which shows 40% performance improvement on average.

Keywords: transactional memory, data reusing, TItDR.

1 Introduction

Recent research has showed that transactional memory has been the promising parallel programming technique. The proposed TM systems (RSTM[1], UTM[2], logTM[3], TCC[4]) must provide atomicity and isolation for transactions. If concurrent executing transactions find that they can not both successfully commit because of conflicting shared memory access, one transaction must be chosen to re-execute. This abort handling is the import part of TM systems as it greatly affects the overall TM system's performance. Next we summarize the well-known published TM systems, focusing on processing if transaction fails. Analysis shows that there exists potential data reusing of the aborted work for failed transaction execution.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 141–150, 2007.

As seen from the currently TM systems(LogTM[3,5,6], TCC[4,7], UTM[2,8], RSTM[1,9]), to abort a transaction, we must do the tedious compensating work to guarantee the transaction's isolation property[10,11,12,1,3]. The burden of memory system doing the restore operation may be greater than the cost of opening the shared object. Because this needs to read and write a large set of dispersed data simultaneously. The retry on failed (i.e. transaction iteration) execution method will reopen the same object with very high chance, the close and then open process is necessary for the conflicting object to ensure consistency. Yet with our experience, other non-conflicting shared object, the reopen process usually does the same thing as reconstruction the metadata, allocating memory space, and updating bookkeeping information etc.

TM system detects conflicts when concurrent transactions visiting the same object and at least one is write[9], reducing transaction conflicts chance can greatly improve the whole system performance. Currently proposed conflicts resolution technique deals with this problem by letting transaction wait for object to be released or back off some time before retry transaction. The principle of reusing data should avoiding introducing more conflicts. Our paper shows that by taking special management, TM system can get this win-win situation.

In this paper, we proposed a common method to reusing transactional object on transaction aborts, which called TItDR. As far as I know, this is the first research on exploring data reusing across transaction iterations. Our contribution includes:

- We proposed a new method called TItDR which exploring the data reusing across transaction iteration.
- We show that current TM system can support our TItDR without redesign from ground.
- We give a hardware framework to support TItDR. Currently proposed transactional protocol can be enhanced to benefit from the data reusing.

2 Basic Idea

In this section, we discuss basic idea of TItDR, and ignore some implementation details which may be different for STM or HTM. In this paper, we call the repeated retry of the transaction until successfully as *transaction iteration*, so one transaction's execution is composed of several iterations with the last iteration is successful; current running iteration is called *active* iteration; the iterations that failed before active iteration is called *obsolete* iteration; the object which has been opened in the active iteration is in *active state*, the object opened in the obsolete iterations but not opened in active iteration is in *obsolete state*. Overall TItDR improved TM system with the following idea:

1. Binding a number called *itnumber* to each transaction iteration, on abort, increase the itnumber and reset itnumber if commits successfully. When shared object is opened, TItDR saves current itnumber together with the object. So by comparing the itnumber with the active iteration's itnumber, we can decide if the object is in obsolete state.

2. Thread maintains a local list of opened objects; on opening shared object, add the object together with current iteration's itnumber to the list. Most TM systems maintain the opened list for rolling back, so we may only a mirror modification. If transaction aborts, we keep the opened list through which TM system keeps information about the obsolete object, and do not try to restore the state of transactional opened object. In the next iteration, if transaction opens an obsolete object, the cost can be greatly reduced because we can reuse the obsolete object. For opening for read an obsolete object, we can use directly use the data if validation is successful, and for opening for write, previous iteration's write data maybe incorrect, so the value should be discarded, but we can reuse the metadata to avoid reconstructing.
3. Other transaction's conflicts with opened object now has two types, conflicts with active objects and conflicts with obsolete objects. Our method keeps the obsolete in the "pseudo-active" state that will enlarge the object's open time. That will introduce more conflicts between transactions. So contention manager should take a compromised decision between these two folds. With our experience, always aborting the obsolete object will not bring more conflicts than current TM systems; more, the obsolete object may not be opened in the re-execution if the execution path is different; so aborting the obsolete object will not introduce some unnecessary conflicts;
4. On reopening object in the re-execution, if object is in obsolete state and the object has not been aborted by other transaction, thread can reuse this object without performing the reopening process.

Our method's primary advantage is it greatly reduces the work needed on transaction abort, because we no longer need to restore memory state before transaction restarts. A second advantage is we reduce the cost of reopening process by reusing obsolete object in active transaction iteration. The reusing includes data value reuse, memory space reuse and data structure reuse.

On read operation, value first read to a temporal local place and uses this value in the remainder transaction, transaction commits failed if object's curren has been updated by other transaction during the transaction. TItDR improves the read operation with transaction's itnumber which will increase by 1 on restarted, if current transaction fails, most current TM systems will discard the read list and temporal read value and start a fresh read operation in the next iteration. Yet we can simply keep the read set and value in the next iteration, on opening an object, if this object is in the obsolete read set(object's itnumber will be less than active iteration's itnumber), we can reuse the value if validation successful. On write operation, new value can be directly updated to the temporal location if previous iteration has opened for write. Our basic idea is simple, we believe that the failed transaction is not having nothing to gain.

As we study from current proposed TM systems, we think that it is feasible to incorporate with our method to have data reusing benefit. In the next section, we give one implementation example based on RSTM and show some problem that is brought with data reusing.

3 Example Software Implementation

STM system usually construct through software library or language extension which uses complex metadata organization[1], in this section, we give the detailed optimization of RSTM to implement our TItDR methods. RSTM is a non-blocking STM system implemented as C++ class library. RSTM support visible/invisible transactional object read, eager/lazy transactional object write. In RSTM, every transactional object is accessed through ObjectHeader, which points to the current version of the object. The ObjectHeader contains the visiting information from transactions; The Transaction Descriptor referenced through an object's header determines the transaction's state. If the transaction commits, then NewDataObject is the current version of the object. If the transaction aborts, then OldDataObject restored to the current version. If the transaction is active, no other transaction can read or write the object without aborting the transaction. We will ignore detailed information in this paper and only gives the optimization of RSTM, which is referred by RSTM_datareuse.

We redefine transaction's ABORTED state, which means the time between current iteration is aborted and next iteration starts. RSTM_datareuse adds itnumber to transaction descriptor to holds transaction's iteration information; ObjectHeader uses the second low-bit of NewData as obsolete flag for eager write, for every entry in the explicit list of visible reader, adds one bit obsolete flag for visible reader. The obsolete flag indicates that object is in obsolete state, which is opened in previous iteration, but has not opened in current iteration.

RSTM uses bookkeeping lists (invsibleReadList, visibleReadList, eagerWriteList, lazyWriteList) to hold currently opened object, In addition RSTM_datareuse adds the list entry with itnumber field which hold current transaction iteration's itnumber.

When transaction initially starts, the bookkeeping lists are empty and the itnumber reset to 1. On read operation, RSTM_datareuse adds the object to thread local invisibleReadList or visibleReadList based on reading type. In case of visible read, mark the corresponding read flag in the ObjectHeader. The read

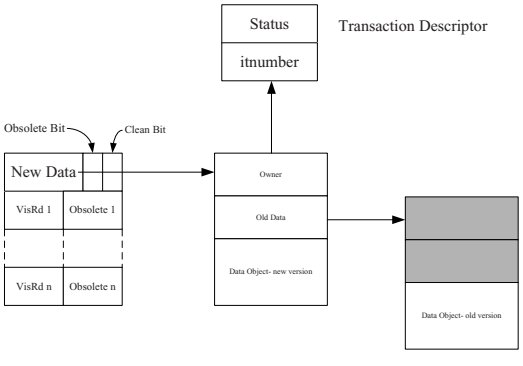


Fig. 1. metadata used to implement TItDR

flag together with the obsolete flag cleared indicating that the read object is in active state. For write operation, RSTM allocate a cloned object to hold new value. The ObjectHeader's owner state with obsolete cleared indicating the object is in write active state. If the transaction is aborted, for visible read and eager write, iterates the list and marks the active object's corresponding obsolete flag. So other transaction may only need to check the obsolete bit to see whether the object is in active state or obsolete state. RSTM_datareuse does not drop the bookkeeping list or free new allocated memory; this has performance benefit as avoids rebuilding the metadata on transaction's re-execution. As comparing with RSTM, RSTM_datareuse's abort processing is really simple.

If transaction is aborted and restarted, now the bookkeeping lists holds opened object in previous iterations. The reopening process makes some difference, first checks if the object is in the bookkeeping list, if found and the itnumber is less than current transaction itnumber, the opened object needs validation to see whether the object is still valid, if validate successfully, clear ObjectHeader's obsolete flag, and now the object is in current transaction's active state. For write operation, because RSTM_datareuse does not free the cloned object, active transaction's cloned object will reuse this memory space. RSTM_datareuse incurs a bit of lookup and validation for open cost, this cost is neglectable because the system needs periodically validating to ensure opened object is still in valid[13]. A pseudo-code for open_RW operation is as follows:

On transaction commit successfully, RSTM_datareuse only update memory of the active objects (i.e. bookkeeping list entry's itnumber equals active iteration's itnumber). There may be obsolete objects in bookkeeping list when transaction commit successfully, this is due to execution path is different between committed iteration and previous iteration. For these objects, we need to restore their previous value before transaction.

The object's obsolete state divides conflicts into two types: transaction conflicts with active object or obsolete object, RSTM_datareuse always abort the obsolete object for several reasons. First, aborting obsolete object's cost is small for it does not need to abort current active transaction execution. Second, releasing obsolete object makes object's open time shorter which allows more transaction parallelism. Third, transaction's open set may be different between iterations if the transaction has branches and the condition is based on the shared object's return value. So always aborting the obsolete object will avoid some false conflicts. The contention management policies can be used with no modification for only conflict with active objects is resolved by CM (Contention Manager) and the introduced obsolete state can be ignored by CM.

Modern STM systems incrementally validate opened object to test whether the execution is valid. RSTM opened set is maintained in transaction's bookkeeping lists, RSTM_datareuse only needs to validate the active object in the list and ignores the obsolete object, because if the obsolete object will be reopened, the opening process includes the validation operation. So although RSTM_datareuse does not drop obsolete objects, its validation cost will not be greater than RSTM.

4 The Hardware Approach

Hardware transactional memory is a hardware system that supports implementing nondurable ACI properties for threads manipulating shared data. A very natural way of implementing HTM is enhancing cache coherence protocol to support transactional processing. LogTM supports eviction of transactional accessed cache lines during a transaction by retaining ownership of the cache line. The cache coherence protocol's directory state is in sticky state when an active transaction's opened object is written back while the ownership is still reserved by transaction. In this way transaction that conflicts with the sticky object can be detected by forwarding the request to the owner if the owner is in transaction mode. A second feature is using software log to restore memory state on abortion. To implement our data reusing idea, we should enhance logTM's protocol with some extensions similar to our software approach.

Transactional object has two copies (*active* copy and *shadow* copy) in cache, the active copy stores current value and shadow copy store backup data. In this way, we no longer need the undo_log maintained by processor, for the backuped value is stored in shadow copy. The space requirement is the mainly cost, yet we think it is acceptable as we can enhance the multi-level cache to support our requirement. Another method to reduce the cost is to use the similar method used in operating system when mapping virtual memory to cache entry mapping. If memory first opened in transaction, update the active and shadow copy with object's current data. The following update to the object will be written to the active copy, if transaction commits successfully, update the memory system with the active copy. On failure, mark active copy as obsolete, if obsolete object is visited, use the shadow copy value.

Transaction's iteration number needs to be hold in processor. Cache and directory maintain iteration number information for every opened object. On transaction fails, processor increase its iteration number and mark opened shared objects as obsolete. On reopening object, the validation processor is really simple, it only needs to see whether the cache block is still valid in cache.

Undo_log: only active transaction's evicted cache object is written into the undo log. If transaction failed and the undo_log is not empty, we should replay the undo_log to restore memory state. The undo_log is managed by software, which dealing the case hardware cache overflows.

Conflicts detection is through the cache coherence protocol. When processor get intervention message which visit the obsolete object in his cache, the processor should forwarding the old data and need not abort current transaction. Another tricky is when another processor conflicts with out-of-cache active objects (i.e. objects in the undo_log), in this situation, simply send NACK message to abort it.

The hardware approach does not need to replay the undo_log to restore memory state if transaction fails. This cost is much greater than the RSTM, because logTM write the previous back to memory, while RSTM simply modify the object's header to point back original data as it keeps both active and backup data in memory. On reopening objects it reuses the shadow copy so saves

memory visiting cost. Currently we are working on the test environment to give our reusing detailed test result.

5 Test Results and Analysis

In this section, we give our test result of data reusing in transactional memory, we implement TItDR based on RSTM2 as shown in section 3. We test benchmarks on a 2-processor blade server with Intel Xeon 2.3GHz, 4core processor. We compiled both our implementation and RSTM with gcc3.4.4 with O3 optimization level. We tested for a period of 10 seconds for each benchmark, varying the thread number from 1 to 8. Results were averaged over a set of 3 test runs and all experiments use the Polka contention manager.

We use the same benchmark with RSTM2[1] which includes: shared counter; linked list; hash table; LFUCache and random graph.

5.1 Total Transaction Throughput

Throughput comparison was given by the total finished transactions during 10 seconds; we give both eager and lazy write type benchmark results.

For the shared counter benchmark, we get the best speedup compared with RSTM2, this is due to that all threads want to increase the same shared variable, which can not be accessed parallel. All the thread must line up to access the counter, so with threads number increase, both our method and RSTM does not increase the total transactions throughput. Yet for RSTM's eager write type, with thread number increase, we got decrease total committed transactions, for it needs more work to contention management, restoring. For the same reason, Eager write type with data reusing will not suffer this problems. Another reason for we get the best speedup is that the counter benchmark is dominated by write, with no read operation. For software approach TM, the reusing for write will save more work than reusing for read. This is due to that we relieve the memory burden of reclaim and reallocate memory space for speculative writes. For cache coherence based HTM systems, it is another case; hardware can easily get his obsolete read object by checking cache status, and return current value.

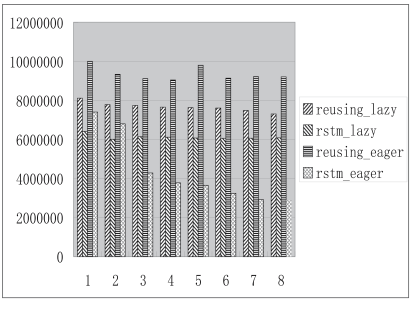


Fig. 2. Throughput of shared counter

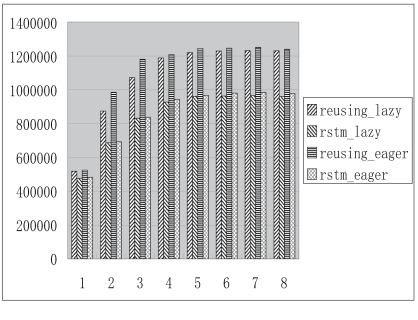


Fig. 3. Throughput of linked list

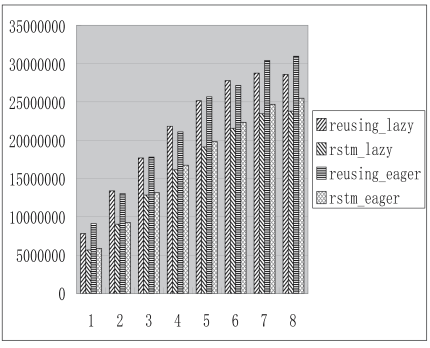


Fig. 4. Throughput of hash table

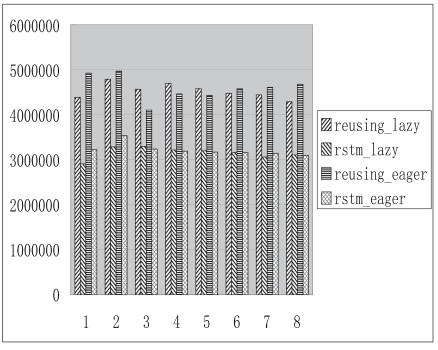


Fig. 5. Throughput of LFUCache

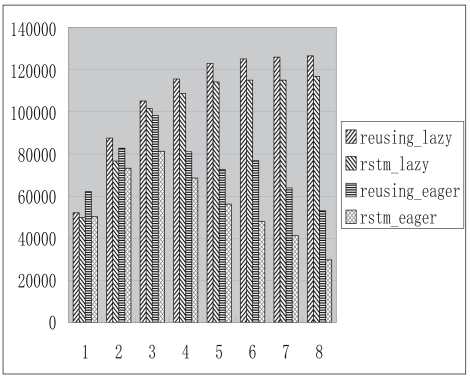


Fig. 6. Throuput of random graph tests

For hash table and random graph benchmarks, we get continued increased throughput as threads number increase, while RSTM's RandomGraph get decreased throughput, it is because that as thread continues add vertex to the graph, the graph will get large, so every time we want to locate a vertex, it must traverse a large number of vertexes before getting to the vertex, this will increase transaction's read and write set and the transaction will getting large.

Table 1. the result is got from running benchmark with 8 threads and uses invisible read, lazy write acquire rule. The number is given on thread average. Validation success on write means that the object has not been updated by other thread. Validation success on read means that our read value is still valid.

Benchmark	W_times	V_success	V_failed	R_times	V_success	V_Failed
Counter	1031806	74566	162	0	0	0
LinkedList	44119	50	1683	12854531	4287378	160397
HashTable	1151057	761	2454	5755649	3920	16814
LFUCache	1218634	134212	3795	115610	351	10390
RandomGraph	123708	183	9406	9314085	2210817	17373

This large transaction's aborting cost is the primary reason for the decreasing performance. With our method, test results show good scalability. Our method's cost of aborting will not change with different transaction size.

TItDR has better performance speedup for eager write type, e.g. the counter benchmark gets 35% to 2.2 times performance increase for eager, and for lazy, we got 25% enhancement. Totally, we got the average performance speedup of 41.4% for all benchmarks.

5.2 Potential of Data Reusing

To see the potential data reusing there exists in transactional memory systems, we count the times of open operation, and the times we can find data in failed transaction's open set, with the times that the data is valid.

For counter and LFUCache benchmark, the benchmark is dominated by write operation, so write conflicts may occur very common, this servers two folds effect, first, it will regularly make other transaction's read not valid, so to reuse read data, there is more chances that validation is failed. Second, the write may have more chances that validation is success. The LinkedList, HashTable and RandomGraph benchmarks are another case; the read operation has more chances to find that the aborted transaction's read value is still valid.

6 Conclusion and Future Work

We believe that TItDR is very attractive for TM systems have great potential data reusing. The TItDR method greatly reduced the work to abort a transaction and will accelerate the reopening process. From my experience, hardware based data reusing is more attractive than software, for it can easy get the real data reusing which can get value from cache. We have worked mainly on the software approach. Yet the software approach should be optimized to efficiently support currently proposed TM systems or rebuild TM system from ground with data reusing in mind. We have not explored how to efficiently reuse data in the nested transaction environment. Further studying cache coherence based transactional memory with data reusing support includes protocol verification, implementation and performance test.

References

1. Marathe, V.J., Spear, M.F., Heriot, C., Acharya, A., Eisenstat, D.: Scherer III, W.N., Scott, M.L.: Lowering the overhead of software transactional memory. Technical Report TR 893, Computer Science Department, University of Rochester (2006)
2. Chuang, W., Narayanasamy, S., Venkatesh, G., Sampson, J., Van Biesbrouck, M., Pokam, G., Calder, B., Colavin, O.: Unbounded page-based transactional memory. In: ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, San Jose, California, USA, pp. 347–358. ACM Press, New York, NY, USA (2006)

3. Moore, K.E., Bobba, J., Moravan, M.J., Hill, M.D., Wood, D.A.: Logtm: Log-based transactional memory. In: Proceedings of the 12th International Symposium on High-Performance Computer Architecture, pp. 254–265 (2006)
4. Hammond, L., Wong, V., Chen, M., Carlstrom, B.D., Davis, J.D., Hertzberg, B., Prabhu, M.K., Wijaya, H., Kozyrakis, C., Olukotun, K.: Transactional memory coherence and consistency. SIGARCH Comput. Archit. News 32(2), 102 (2004)
5. Liblit, B.: An operational semantics for LogTM. Technical Report, University of Wisconsin–Madison (2006) Version 1.0 (1571)
6. Moore, K.E.: Thread-level transactional memory. In: Wisconsin Industrial Affiliates Meeting (2004)
7. Hammond, L., Carlstrom, B.D., Wong, V., Hertzberg, B., Chen, M., Kozyrakis, C., Olukotun, K.: Programming with transactional coherence and consistency (tcc). In: ASPLOS-XI: Proceedings of the 11th international conference on Architectural support for programming languages and operating systems, pp. 1–13. ACM Press, New York, NY, USA (2004)
8. Lie, S.: Hardware support for unbounded transactional memory. Master's thesis, Massachusetts Institute of Technology (2004)
9. William, N., Scherer, I., Scott, M.L.: Advanced contention management for dynamic software transactional memory. In: PODC 2005. Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing, pp. 240–248. ACM Press, New York, NY, USA (2005)
10. Herlihy, M., Moss, J.E.B.: Transactional memory: architectural support for lock-free data structures. In: ISCA 1993. Proceedings of the 20th annual international symposium on Computer architecture, pp. 289–300. ACM Press, New York, NY, USA (1993)
11. Herlihy, M., Luchangco, V., Moir, M.: A flexible framework for implementing software transactional memory. In: OOPSLA 2006. Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, pp. 253–262. ACM Press, New York, NY, USA (2006)
12. Saha, B., Adl-Tabatabai, A.-R., Hudson, R.L., Minh, C.C., Hertzberg, B.: Mcrt-stm: a high performance software transactional memory system for a multi-core runtime. In: PPoPP 2006. Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming, pp. 187–197. ACM Press, New York, NY, USA (2006)
13. Spear, M.F., Marathe, V.J., Scherer III, W.N., Scott, M.L.: Conflict detection and validation strategies for software transactional memory. In: DISC, pp. 179–193 (2006)

A Parallel BSP Algorithm for Irregular Dynamic Programming

Malcolm Yoke Hean Low[1], Weiguo Liu[1], and Bertil Schmidt[2]

[1] School of Computer Engineering, Nanyang Technological University, Singapore 639798
{yhlow, liuweiguo}@ntu.edu.sg
[2] University of New South Wales Asia, 1 Kay Siang Road, Singapore 248922
bertil.schmidt@unswasia.edu.sg

Abstract. Dynamic programming is a widely applied algorithm design technique in many areas such as computational biology and scientific computing. Typical applications using this technique are compute-intensive and suffer from long runtimes on sequential architectures. Therefore, several parallel algorithms for both fine-grained and coarse-grained architectures have been introduced. However, the commonly used data partitioning scheme can not be efficiently applied to irregular dynamic programming algorithms, i.e. dynamic programming algorithms with an uneven load density pattern. In this paper we present a tunable parallel Bulk Synchronous Parallel (BSP) algorithm for such kind of applications. This new algorithm can balance the workload among processors using a tunable block-cyclic data partitioning method and thus is capable of getting almost linear performance gains. We present a theoretical analysis and experimentally show that it leads to significant runtime savings for pairwise sequence alignment with general gap penalties using BSPonMPI on a PC cluster.

Keywords: BSP, Irregular Dynamic Programming, Partitioning, Load Balancing, Scientific Computing.

1 Introduction

Dynamic programming (DP) is a popular algorithm design technique for optimization problems. Problems such as string editing [1], genome sequence alignment [14, 22], RNA and protein structure prediction [6, 17, 24], context-free grammar recognition [7, 19], and optimal static search tree construction [9] have efficient sequential DP solutions. In order to reduce the high computing cost of DP problems, many efficient parallel algorithms on different parallel architectures have been introduced [1, 2]. On fine-grained architectures, the computation of each cell within an anti-diagonal is parallelized [20, 21]. However, this way is only efficient on architectures such as systolic arrays, which have an extremely fast inter-processor communication. On coarse-grained architectures like PC clusters it is more convenient to assign an equal number of adjacent columns to each processor as shown in Figure 1. In order to reduce communication time further, matrix cells can be grouped into blocks.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 151–160, 2007.

Processor P_i then computes all the cells within a block after receiving the required data from processor P_{i-1}. Figure 1 shows an example of the computation for 4 processors, 8 columns and a block size of 2×2, the numbers 1 to 7 represent consecutive phases in which the cells are computed. We call this method *blockbased*. It works efficiently for regular DP computations with an even workload across matrix cells, i.e. each matrix cell is computed from the same number of other matrix cells.

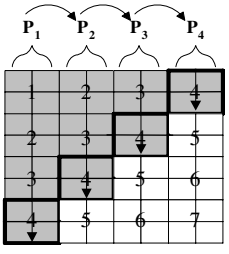


Fig. 1. Parallel computation for 4 processors, 8 columns and a 2×2 block size

In practice, there are many irregular DP applications where the workload of a cell varies across the matrix. Figure 2 shows an example of such an application. The workload to compute one matrix cell will increase along the shift direction of the computation. We call this the *load computation density*. Figure 2 shows the change of load computation density along the computation shift direction by using increasingly blacking shades. We can see that the load computation density at the bottom right-hand corner is much higher than that in the top left-hand corner. The column-based partitioning method in Figure 1 will therefore lead to a poor performance, since the workload on processor P_i is much higher than on the processor P_{i-1}.

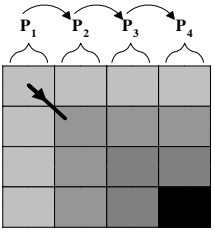


Fig. 2. Example of an irregular DP computation

In this paper, we propose a general parameterized parallel BSP algorithm to solve this problem. By introducing two performance-related parameters, we can get the trade-off between load balancing and communication time by tuning these two parameters and thus obtain the maximum possible performance. We demonstrate how this algorithm can lead to substantial performance gains for irregular DP applications.

The rest of the paper is organized as follows: Section 2 describes the characters and classification for irregular DP algorithms. The BSP model is briefly reviewed in Section 3. Section 4 presents the parallel BSP algorithm. Section 5 evaluates the performance on a PC clusters using BSPonMPI. Section 6 concludes this paper.

2 Irregular DP Algorithms

DP algorithms can be classified according to the matrix size and the dependency relationship of each matrix cell [10]: a DP algorithm for a problem of size n is called a *tD*/*eD* algorithm if its matrix size is $O(n^t)$ and each matrix cell depends on $O(n^e)$ other cells. The DP formulation of a problem always yields an obvious algorithm whose time complexity is determined by the matrix size and the dependency relationship. If a DP algorithm is a *tD*/*eD* problem, it takes time $O(n^{t+e})$ provided that the computation of each term takes constant time. Three examples are given in Algorithm 1 to 3.

Algorithm 1. (2*D*/0*D*): Given $D[i,0]$ and $D[0, j]$ for $1 \le i, j \le n$,
$D[i, j] = \min\{D[i\text{-}1, j] + x_i, D[i, j\text{-}1] + y_j, D[i\text{-}1, j\text{-}1] + z_{i,j}\}$ where x_i, y_j and $z_{i,j}$ are computed in constant time.

Algorithm 2. (2*D*/1*D*): Given $w(i, j)$ for $1 \le i<j \le n$; $D[i,i] = 0$ for $1 \le i \le n$

$$D[i,j] = w(i, j) + \min_{i<k\le j} \{D[i,k-1] + D[k, j]\} \quad \text{for } 1 \le i, j \le n$$

Algorithm 3. (2*D*/2*D*): Given $w(i, j)$ for $1 \le i<j \le 2n$; $D[i, 0]$ and $D[0, j]$ for $0 \le i, j \le n$,

$$D[i, j] = \min_{\substack{0\le i'<i \\ 0\le j'<j}} \{D[i',j'] + w(i'+j',i+j)\} \quad \text{for } 1 \le i, j \le n$$

Table 1. A classification for the popular DP algorithms in CB

Algorithm	Time complexity	Application Field	Reference
Smith-Waterman algorithm with linear and affine gap penalty	$O(n^2)$	Genome alignment	[14, 22]
Syntenic alignment		Generalized genome global alignment	
Smith-Waterman algorithm with general gap penalty	$O(n^3)$	Genome alignment	[8, 22]
Nussinov algorithm		RNA base pair maximization	
Viterbi Algorithm	$O(n^2)$~$O(n^4)$	Gene sequence alignment using HMMs, Multiple sequence alignment	[8]
Double DP algorithm	$O(n^4)$	Protein threading	[17]
Spliced Alignment	$O(n^3)$	Gene finding	[11]
Zuker Algorithm	$O(n^3)$~$O(n^4)$	RNA secondary structure prediction	[24]
CYK Algorithm		RNA secondary structure alignment	[8]

There are many DP algorithms in Computational Biology (*CB*). DP is used for assembling DNA sequence data from the fragments that are delivered by automated sequencing machines [3], and to determine the intron/exon structure of eukaryotic genes [12]. It is used to infer function of proteins by homology to other proteins with known function [18, 23] and it is used to predict the secondary structure of functional RNA genes or regulatory elements. In some areas of CB, DP problems arise in such

variety that a specific code generation system for implementing such algorithms has been developed [4]. However, the development of a successful parallel DP algorithm is a matter of experience, talent, and luck. The typical matrix recurrence relations that make up a parallel DP algorithm are intricate to construct, and difficult to implement reliably. No general problem independent guidance is available. Table 1 shows the classification of some popular DP algorithms in CB.

(a) Nussinov: Given a sequence A of length L with symbols $x_{1,\ldots}\, x_L$. Let $\delta(i, j) = 1$ if x_i and x_j are a complementary base pair, else $\delta(i, j) = 0$. We will recursively calculate scores $M(i, j)$ which are the maximal number of base pairs that can be formed for subsequence $x_i, \ldots, x_j$.

Initialization:
for $i = 2$ **to** L **do**
  $M(i,i-1) = 0$
for $i = 1$ **to** L **do**
  $M(i,i) = 0$

Recursion:
$M(i, j) = \max\{M(i+1, j), M(i, j-1),$
$M(i+1, j-1) + \delta(i, j),$
$\max_{i<k<j}[M(i, k)+M(k+1, j)]\}$

(b) SkylineMatrix: The skyline matrix problem can be formulated as follows: Given an $N \times N$ skyline matrix A and an N-vector b, we seek to find an N-vector x such that $Ax = b$. An efficient and widely used technique for solving $Ax = b$ in the general case is the LU-Decomposition. This method decomposes A into two matrices L and U. The algorithm used for sequential LU-Decomposition is "Doolittle's Method". Generally, the algorithm works as follows:

for $i = 1$ **to** N **do**
  for $j = 1$ **to** $i-1$ **do**
    $L_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj}\right) / U_{jj}$
  for $j = 1$ **to** i **do**
    $U_{ji} = a_{ji} - \sum_{k=1}^{j-1} L_{jk} U_{ki}$

(c) SW with general gap penalty function: Consider two strings A and B of length l_1 and l_2, a substitution matrix s and a general gap penalty function $\gamma(g)$. To identify common subsequences, they compute the similarity matrix $M(i, j)$ of two sequences ending at position i and j.

$$M(i,j) = \max \begin{cases} M(i-1, j-1) + s(A_i, B_j), \\ M(k, j) + \gamma(i-k), \quad k = 0,\ldots,i-1, \\ M(i,k) + \gamma(j-k), \quad k = 0,\ldots,j-1. \end{cases}$$

Fig. 3. The recurrence formulas for three 2D/1D DP algorithms: (a) Nussinov algorithm, (b) Skyline matrix problem, (c) Smith-Waterman algorithm with general gap penalty function

In this paper we concentrate on the parallelization of DP algorithms of the type 2D/1D. This is an important DP algorithm with many applications. Figure 3 shows three well-known DP algorithms of type 2D/1D. Although these DP algorithms look different, they share similar characteristics. These 2*D*/1*D* DP algorithms are all irregular with load computation density changes along the computation shift direction.

Figure 4 shows the change of load computation density along the computation shift direction by using increasingly blacking shades. For these algorithms, the column-based partitioning method of Figure 1 leads to poor load balancing. Thus, a more efficient data partitioning scheme is needed. The problem of determining an appropriate data partitioning scheme is to maximize system performance by balancing the computational load among processors. Since the data partitioning scheme largely determines the performance and scalability of a parallel algorithm, a great deal of research has aimed at studying different data partitioning schemes. As a result the

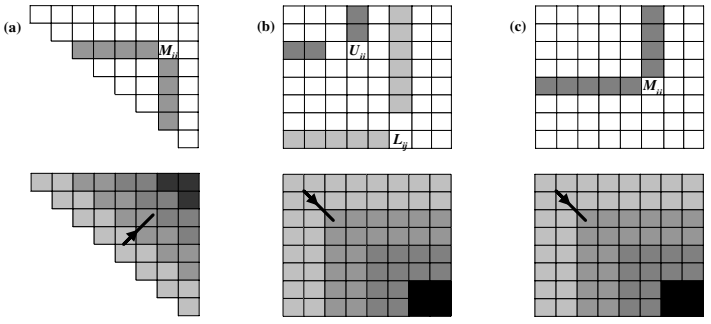


Fig. 4. Dependency relationship and distribution of load computation density along computation shift direction for (a) Nussinov, (b) Skyline matrix problem, (c) Smith-Waterman algorithm with general gap penalty function

block-cyclic partitioning has been suggested as a general-purpose basic scheme for parallel algorithms because of its scalability, load balancing and communication properties [15]. In this paper, we introduce a tunable block-cyclic based distribution of columns for irregular DP algorithms to balance the workload among processors.

3 The Bulk Synchronous Parallel (BSP) Model

The BSP model first proposed in [23] is designed to be a general purpose approach to parallel computing that allows the separation of concerns between computation, synchronization and communication costs. It has a simple cost model for predicting the performance of BSP algorithms on different parallel platforms. A BSP programming model consists of P processors linked by an inter-connecting network and each with its own pool of memory.

A BSP algorithm consists of a set of processors each executing a series of supersteps. Each superstep consists of three ordered phases: 1) a local computation phase, where each processor can perform computation using local data and issue communication requests; 2) a global communication phase, where data is exchanged between processors according to the requests made during the local computation phase; and 3) a barrier synchronization, which waits for all data transfers to complete and makes the transferred data available to the processors for use in the next superstep. The BSP cost model for a BSP algorithm S can be expressed as

$$\text{cost(S)} = \text{sum}\{\ w(i) + gh(i) + L\ \} \text{ for superstep } i = 1 \ \ldots\ n_s$$

where n_s is the total number of supersteps; $w(i)$ is the maximum computation cost by any processor in superstep i; and $h(i)$ is the maximum number of messages sent or received respectively by any processor in superstep i. The architecture dependent parameters g and L represent the communication and synchronization costs respectively. From the BSP cost model, we can see that the performance of a BSP algorithm relies on three factors: a) computation balance; b) communication balance; and c) n_s, the total number of supersteps.

While a BSP library consists of a small set of architectural independent programming interface that support the BSP programming model, the efficiency of a BSP algorithm depends on how the underlying BSP library implementation optimizes the architecture dependent parameters *g* and *L*. Existing BSP library implementation such as the Oxford BSP library [13] and the Paderborn University BSP (PUB) Library [5] are often optimized for a selection of parallel hardware platforms. To keep up with changes and development in these platforms, these libraries have to be constantly updated. The BSPonMPI library (http://bsponmpi.sourceforge.net) is an effort to create a BSP library that runs on any machine that has MPI installed. This ensures that any BSP program compiled using BSPonMPI will benefit from improvements and optimizations in the MPI library for a particular hardware platform.

4 Parallel BSP Algorithm

In this section, we describe a tunable parallel BSP algorithm for solving irregular DP problems. The algorithm proceeds in a series of wavefront diagonally across the matrix *M*. Figure 5 illustrate the concept of the algorithm for an 8×8 matrix with a column-wise block-cyclic partition. The parameter *division* is used to implement a block cyclic distribution of columns to processors. The parameter *rowwidth* is used to control the size of messages that P_i will send to other processors. In the figure, $P_{i,dj}^{\ k}$ denotes that the cell is updated by processor P_i at division *j* of wavefront *k*. Each wavefront corresponds to a superstep in the BSP computation. For example, in wavefront 4, processor P_1 and P_2 are active in both division 1 and 2.

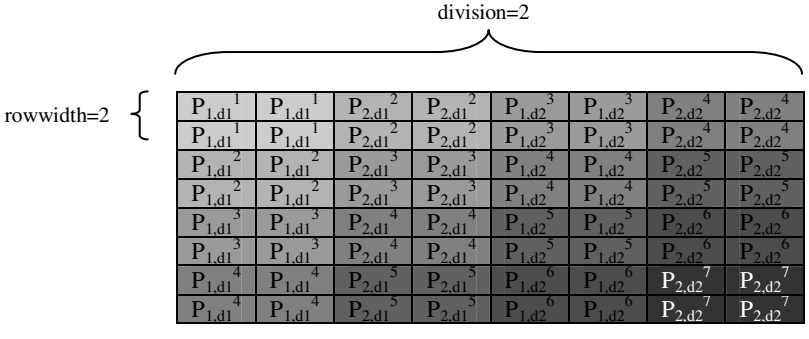


Fig. 5. The tunable block-cyclic partitioning method for irregular dynamic programming

Increasing the number of cyclic divisions and decreasing the size of messages may lead to better load balancing at the expense of increase in communication overhead. Thus, the choice of the parameter for *division* and *rowwidth* is a trade-off between load balancing and communication time. Figure 6 shows the BSP algorithm for irregular dynamic programming. In each superstep (or wavefront), each processor updates the block allocated to it in all its active divisions and sends the updated block to other processors. In this implementation, we use the BSP shared memory primitive `bsp_put()` to update the matrix block. A barrier synchronization is called at the end of each superstep. All processors will receive the updated matrix by the beginning

```
Input:   The number of processors Np, the number of division Nd, the
         row width R. (n×n is the size of matrix M, dt denotes the t-
         th division, wt denotes the t-th wave, C denotes the column
         width).
Output:  Depending on the requirements of the given applications,
         the output will be the optimal score M[1,n] or the whole
         matrix M.

Nwaves = p*Nd + n/R;
C = n/Nd;

bsp_begin(Np)
  pid = bsp_pid();
  // beginning of a superstep, do for each wavefront
  for wt = 1 to Nwaves
    for dt = 1 to Nd  // do for each division
      // if processor pid is active in this division
      if pid+(dt × Np) <= wt
        Sc = (pid + (dt-1)×Np) × C; // compute starting column
        Sr = (wt - pid -(dt × Np)) × R; // compute starting row
        for i = Sc to Sc + C
          for j = Sr to Sr + R
          compute(M[i,j]);
          endfor
        endfor
        send_block();//send updated block to other processors
      endif
    endfor
    // end of superstep
    bsp_sync();
  endfor
bsp_end()
```

Fig. 6. The BSP algorithm for irregular dynamic programming

of the next superstep. Note that for sake of simplicity, the algorithm presented assumes the dimension of the matrix n is exactly divisible by C and R. The actual algorithm implemented does not have this assumption.

5 Performance Evaluation

We carried out a set of experiments using the BSP algorithm described in section 4 to parallelize the Smith-Waterman algorithm with general gap penalty function. The hardware platform used is an 8-node Dual-Processor Linux cluster with a 1GBit/sec Myrinet switch used as inter-cluster connection. The BSP algorithm is compiled with Myrinet MPICH ver 1.2.6 and linked with the BSPonMPI ver 2.0 library.

Table 2 shows the speedup results using the BSP algorithm for irregular dynamic programming on different number of processors. With different number of processors, the best speedup (shown in bold) is obtained with different combination of N_d and R.

In the first implementation, each processor is allocated equal number of columns in each division. When the dimension of the matrix is not exactly divisible by the N_d and the number of processor, the remainder columns are allocated to the first processor in the first division. For example, in the case of N_p=16 and N_d=50, each processor will be allocated 3 columns in each division. In the first division, processor P_1 will be allocated the remaining 600 columns in addition to the 3 columns allocated to each processor! Since processor P_1 will be active in division 1 for n/R supersteps, this

allocation will result in computation and communication imbalance during the BSP computation. Table 3 shows the number of extra columns allocated to processor P_1 in division 1. Except for N_p=2, the best speedup numbers from Table 2 clearly matches the value of N_d that gives the smallest number of extra columns in division 1.

In the second implementation, a more balanced partitioning approach is used. In this implementation, all processors are allocated $k = n/(N_pN_d)$ columns in all divisions. If N_pN_d does not divide n exactly, in division 1, the remaining $n-kN_p(N_d-1)$ columns are divided again equally among all processors and the remaining columns are allocated to P_1. For example, in the case of N_p=16 and N_d=50, each processor will be allocated 3 columns in each division except division 1. In division 1, each processor

Table 2. Speedup for N_d=50 to 90 and row width R=10 to 40 with N_p=2, 4, 8 and 16 processors. The DP matrix is of size 3000×3000.

	N_d=50	60	70	80	90		N_d=50	60	70	80	90
	N_p=2						N_p =4				
R=10	1.56	1.55	1.65	1.59	1.43		2.39	2.76	2.72	2.60	2.92
20	1.63	1.58	1.62	1.49	1.41		2.93	2.20	2.79	2.60	2.74
30	1.46	1.47	1.55	1.57	1.53		**3.15**	2.79	2.69	2.54	2.84
40	1.50	1.61	1.40	**1.66**	1.60		2.43	2.96	2.63	2.60	2.71
	N_p =8						N_p =16				
R=10	4.46	4.56	4.09	3.43	4.71		3.46	3.81	2.55	3.75	4.74
20	3.77	3.98	4.38	3.48	4.68		3.57	4.42	2.96	3.57	**6.60**
30	4.13	**4.94**	4.45	3.13	4.92		3.27	5.81	2.70	3.26	6.47
40	4.37	4.77	4.18	3.45	4.63		3.15	6.10	2.49	3.59	5.77

Table 3. Number of extra columns allocated to processor 1 in division 1

N_d=50	60	70	80	90		N_d=50	60	70	80	90
N_p =2						N_p =4				
0	0	60	120	120		0	120	200	120	120
N_p =8						N_p =16				
200	120	200	440	120		600	120	760	440	120

Table 4. Speedup using BSP algorithm for N_d=50 to 90 and row width R = 10 to 40 with N_p= 2, 4, 8 and 16 processors using improved partitioning. The DP matrix is of size 3000×3000.

	N_d=50	60	70	80	90		N_d=50	60	70	80	90
	N_p =2						N_p =4				
R=10	1.59	1.58	1.64	1.60	1.60		2.82	3.16	3.04	2.92	**3.24**
20	1.62	1.56	1.59	1.62	1.64		2.65	2.99	3.09	3.11	3.20
30	1.44	1.64	**1.66**	1.56	1.71		3.17	2.52	2.67	2.99	2.68
40	1.58	1.61	1.61	1.62	1.66		3.12	2.55	2.97	2.31	3.08
	N_p =8						N_p =16				
R=10	**5.98**	5.78	5.52	5.23	5.64		9.09	7.40	7.12	8.14	8.33
20	5.62	5.79	5.84	5.15	5.62		7.58	**9.36**	7.43	8.70	8.19
30	4.96	5.79	5.58	5.05	5.41		6.65	5.40	7.98	6.34	7.68
40	5.45	5.20	5.17	5.18	5.20		8.02	7.68	3.77	3.76	6.59

will be allocated 40 columns each and processor 1 will receive 48 columns. Another alternative partitioning approach is to allocate the remaining columns equally across all divisions. This will be investigated in our future implementation.

Table 4 shows the experimental results using the improved partitioning. For N_p=16, there is clearly a substantial improvement in performance and the difference in performance between different N_d is reduced. The results show that a balanced partitioning approach is crucial to the performance of the BSP algorithm for irregular dynamic programming.

6 Conclusions and Future Work

In this paper, we have described a tunable BSP algorithm for irregular DP algorithms of type 2D/1D. In the BSP algorithm presented in Figure 6, communication is initiated through the BSP shared memory primitive `bsp_put()` invoked by each sender processor. The receiving part of the communication is automatically handled by the BSPonMPI library and is carried out in bulk at the end of every superstep. This makes the code simple and easy to understand. Note that such one-sided communication primitive is also available in MPI 2.0. An MPI algorithm for irregular DP applications similar to the one presented in [16] that uses matching *send* and *receive* primitive for inter-processor communication can sometime lead to code that is hard to understand and debug.

The experimental results also show that good partitioning approach is essential to achieving high parallel efficiency for this BSP algorithm. The corresponding parallel efficiency for P = 2, 4 and 8 ranges from 75% to 83%. For P = 16, the parallel efficiency drops to 58%. Table 4 shows that the selection of N_d and R has a more significant effect on the performance N_p=16. This could be due to (1) the relatively high barrier synchronization cost L for 16 processors; and (2) the scheduling of tasks between each of the two processors in each node of the Linux cluster.

With improved performance of future versions of the BSPonMPI library, the effects of barrier synchronization cost will be minimized accordingly. We will explore different processor mapping and data partitioning strategies to resolve the issue of scheduling dual-processor nodes in a cluster. Our future work also includes benchmarking the communication and synchronization cost of different processor configurations for our system. This will allow us to predict the performance of different combinations of N_d and R and determine the combination that will yield the best performance. We will also explore how the BSP algorithm can be adapted to other type of DP applications such as 2D/2D and 3D/1D. Such applications are frequently used in the field of computational biology.

References

1. Alves, C.E.R., Cáceres, E.N., Dehne, F.: Parallel dynamic programming for solving the string editing problem on a CGM/BSP. In: Proc. of the fourteenth annual ACM symposium on Parallel algorithms and architectures, Winnipeg, Manitoba, Canada (2002)
2. Alves, C.E.R., Cáceres, E.N., Dehne, F., Song, S.W., Parallel, A.: Wavefront Algorithm for Efficient Biological Sequence Comparison. In: Kumar, V., Gavrilova, M., Tan, C.J.K., L'Ecuyer, P. (eds.) ICCSA 2003. LNCS, vol. 2667, pp. 249–258. Springer, Heidelberg (2003)

3. Anson, E.L., Myers, G.W.: Realigner: A Program for Refining DNA Sequence Multi-Alignments. In: 1st Conference on Computational Molecular Biology, pp. 9–16 (1997)
4. Birney, E., Durbin, R.: Dynamite: A Flexible Code Generating Language for Dynamic Programming Methods. In: Proc. Intelligent Systems for Molecular Biology, pp. 56–64 (1997)
5. Bonorden, O., Juurlink, B., von Otte, I., Rieping, I.: The Paderborn University BSP (PUB) Library. Parallel Computing 29(2), 187–207 (2003)
6. Bowie, J., Luthy, R., Eisenberg, D.: A Method to Identify Protein Sequences That Fold Into A Known Three-dimensional Structure. Science 253, 164–170 (1991)
7. Ciressan, C., Sanchez, E., Rajman, M., Chappelier, J.C.: An FPGA-based coprocessor for the parsing of context-free grammars. In: IEEE Symposium on Field-Programmable Custom Computing Machines (April 2000)
8. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological Sequence Analysis-Probabilistic Models of Protein and Nucleic Acids. Cambridge University Press, Cambridge (1998)
9. Farach, M., Thorup, M.: Optimal evolutionary tree comparison by sparse dynamic programming. In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, November 20-22, 1994, pp. 770–779 (1994)
10. Galil, Z., Park, K.: Dynamic Programming with Convexity, Concavity and Sparsity. Theoretical Computer Science 92, 49–76 (1992)
11. Gelfand, M.S., Mironov, A.A., Pevzner, P.A.: Gene Recognition Via Spliced Sequence Alignment. Proc. Natl. Acad. Sci. 93, 9061–9066 (1996)
12. Gelfand, M.S., Roytberg., M.A., Dynamic, A.: Programming Approach for Prediction the Exon-Intron Structure. Biosystems 30, 173–182 (1993)
13. Hill, J., McColl, B., Stefanescu, D., Goudreau, M., Lang, K., Rao, S., Suel, T., Tsantilas, T., Bisseling, R.: BSPlib: The BSP programming library. Parallel Computing 24(14), 1947–1980 (1998)
14. Huang, X., Chao, K.M.: A Generalized Global Alignment Algorithm. Bioinformatics 19(2), 228–233 (2003)
15. Kumar, V., Grama, A., Gupa, A., Karypis, G.: Introduction to Parallel Computing. Cummings Publishing Company Inc., The Benjamin (1994)
16. Liu, W., Schmidt, B.: A Tunable Coarse-Grained Parallel Algorithm for Irregular Dynamic Programming Applications. In: Bougé, L., Prasanna, V.K. (eds.) HiPC 2004. LNCS, vol. 3296, Springer, Heidelberg (2004)
17. Mount, D.W.: Bioinformatics-Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press (2001)
18. Needleman, S.B., Wunsch., C.D., General, A.: Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. J. Mol. Biol. 48, 443–453 (1970)
19. Ney, H.: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. IEEE Trans. on Acoustic, Speech and Signal Processing ASSP-32(2), 263–271 (1984)
20. Schmidt, B., Schroder, H., Schimmler, M.: Massively Parallel Solutions for Molecular Sequence Analysis. In: Proc. of IPDPS 2002 (2002)
21. Schmidt, B., Schroder, H., Schimmler, M.: A Hybrid Architecture for Bioinformatics. Future Generation Computer System 18, 855–862 (2002)
22. Smith, T.F., Waterman, M.S.: Identification of Common Subsequences. Journal of Molecular Biology 147, 195–197 (1981)
23. Valiant, L.G.: A Bridging Model for Parallel Computation. Communications of the ACM 33(8), 103–111 (1990)
24. Zuker, M., Stiegler, P.: Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information. Nucleic Acids Research, 9 (1981)

Context-Aware Middleware Support for Component Based Applications in Pervasive Computing*

Di Zheng, Yan Jia, Peng Zhou, and Wei-Hong Han

School of Computer Science,
National University of Defence Technology,
Changsha, Hunan, China 410073
dizheng@nudt.edu.cn

Abstract. Ubiquitous computing allows application developers to build a large and complex distributed system that can transform physical spaces into computationally active and intelligent environments. Ubiquitous applications need a middleware that can detect and act upon any context changes created by the result of any interactions between users, applications, and surrounding computing environment for applications without users' interventions. The context-awareness has become the one of core technologies for application services in ubiquitous computing environment and been considered as the indispensable function for ubiquitous computing applications. The need for high quality context management is evident to the component-based middleware for it forms the basis of the component adaptation and the component deployment in the pervasive computing. Therefore, we suggest a holistic approach where context management is an integral part of a more comprehensive adaptation enabling middleware, thus enabling the development and support of context-aware, component-based applications.

Keywords: Pervasive Computing, Context Management, Middleware, Component.

1 Introduction

With the technical evolution of wireless networks, mobile and sensor technology, the vision of pervasive computing is becoming a reality. The paradigm for pervasive computing aims at enabling people to contact anyone at anytime and anywhere in a convenient way. And it also brings the new challenges to traditional applications [1, 2]. In this environment, the applications should become context aware for the reason that the resources (e.g. memory, battery, CPU) of the mobile devices may be limited and the application execution context (e.g. user location, device screen size) is variable[3].Therefore, the applications need adapt their behaviors basing on corresponding context information. The context-awareness has become the one of

* This work was funded by the National Grand Fundamental Research 973 Program of China under Grant No.2005cb321804, the National High-Tech Research and Development Plan of China under Grant No.2004AA112020.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 161–171, 2007.

core technologies for application services in ubiquitous computing environment and been considered as the indispensable function for ubiquitous computing applications.

At the same time, middleware is a widely used term to denote generic infrastructure services above operating system and protocol stack. The role of the middleware is to ease the task of designing, programming and managing distributed applications by providing a simple, consistent and integrated distributed programming environment. As the name ”middleware” suggests, it uses the hierarchical model, a layer of services which sits between applications and operating systems and it can allow the clients to invoke the operations on distributed objects without concern for object location, programming language, OS platform, communication protocols and interconnects, and hardware. Therefore, ubiquitous applications need a middleware that can detect and act upon any context changes created by the result of any interactions between users, applications, and surrounding computing environment for applications without users' interventions.

In order to provide context-awareness services, the middleware platform supporting ubiquitous computing should be able to recognize contextual changes so that applications use contexts for evaluating new environments and finding an appropriate action by the result of evaluation for these changes. Furthermore, many of the current applications are component-based and the management of the components in the pervasive computing environment will be more difficult than ever. However, the existing component based middleware (e.g., .NET[4], Enterprise JavaBeans [5], and the CORBA Component Model [6]) do not take charge of the context information. The OMG Specification of the Deployment and Configuration [7] and the deployment tool of COACH [8] only support the context that presents the deployment target environment and do not consider the context in general case. Therefore, we put forward a context-aware middleware to support QoS-aware context management, context-aware adaptations and deployment for component-based pervasive applications in this paper and discussed according mechanisms.

2 Architecture of the Context-Aware Middleware for Component-Based Pervasive Computing

2.1 Component Based Middleware StarCCM

In terms of middleware, lots of emphasis has been given to enterprise (or server-side) component technologies, such as Enterprise Java Beans or the CORBA Component Model. As depicted in figure 1, in previous work we have developed a component based middleware StarCCM which conform to the CORBA Component Model. The components execute inside a container, which provides implicit support for distribution in terms of support for transactions, security, persistence and resource management. This offers an important separation of concerns in the development of business applications; i.e. the application programmer can focus on the development and potential re-use of components to provide the necessary business logic, and a more “distribution-aware” developer can provide a container with the necessary non-functional properties. Containers also provide additional functionality including life-cycle management and component discovery.

The OMG Specification of the Deployment and Configuration presents a data model for the description of a deployment plan which contains information about artifacts that are part of the deployment, how to create component instances from artifacts, where to instantiate them, and information about connections between them. This specification also presents a data model for the description of the domain into which applications can be deployed as a set of inter-connected nodes with bridges routing between inter-connects. However, these data models are still insufficient for the context of the mobile devices and do not support a description of the rules achieving the adaptation of the deployment.

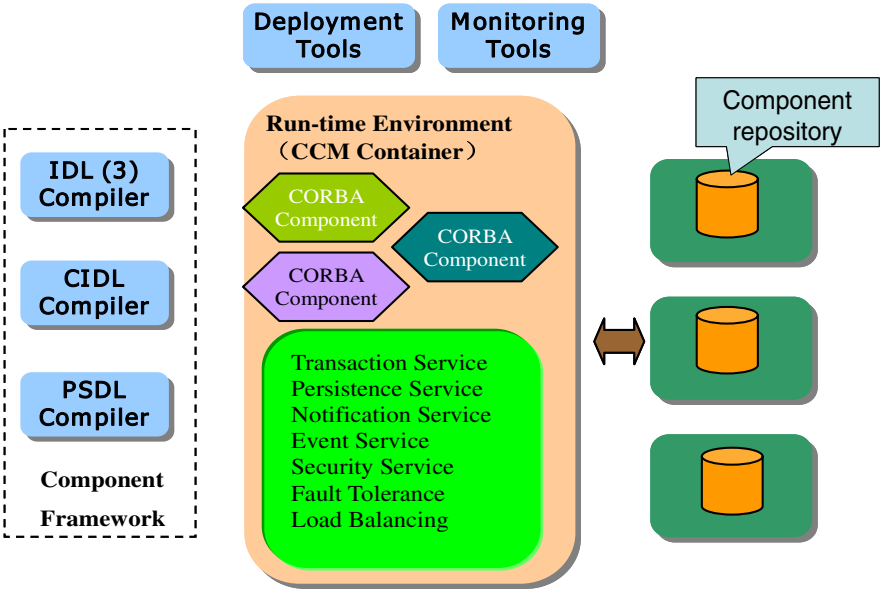


Fig. 1. The Architecture of the StarCCM

2.2 Architecture of Component Based Context-Aware Middleware

The overall middleware architecture is shown in the figure 2. The core provides the fundamental platform-independent services for the management of applications, components and component instances. The core relies on the basic mechanisms for instantiation, deployment and communication provided by the distributed computing environment.

StarCCM Core Component Management provides platform-independent services for the management of the component based applications, components and component instances as depicted in the figure 1. It also provides uniform platform-independent access to the execution platform resources. Furthermore, the middleware offers the other three core services:

- The Context manager which monitors the user and the execution context for detection of relevant changes.
- The Adaptation Manager which reasons about the impact of the changes and decides about appropriate adaptations based on architectural description of component properties.
- The Configurator which reconfigures the application variant to put the decided adaptations into effect.

Context Manager is responsible for sensing and capturing context information and changes, providing access to context information (pull) and notifying context changes (push) to the Adaptation Manager. The Context Manager is also responsible for storing user needs and preferences on application services. The Context Manager should provide flexible context sensing. We recommend the Context Manager to be developed as a Component Framework where new context sensor components can be plugged in. The Context Manager may provide advanced reasoning operations on context. For example, it may aggregate complex context elements from elementary context elements or derive user needs from context. The Context Manager may also keep track of context change history.

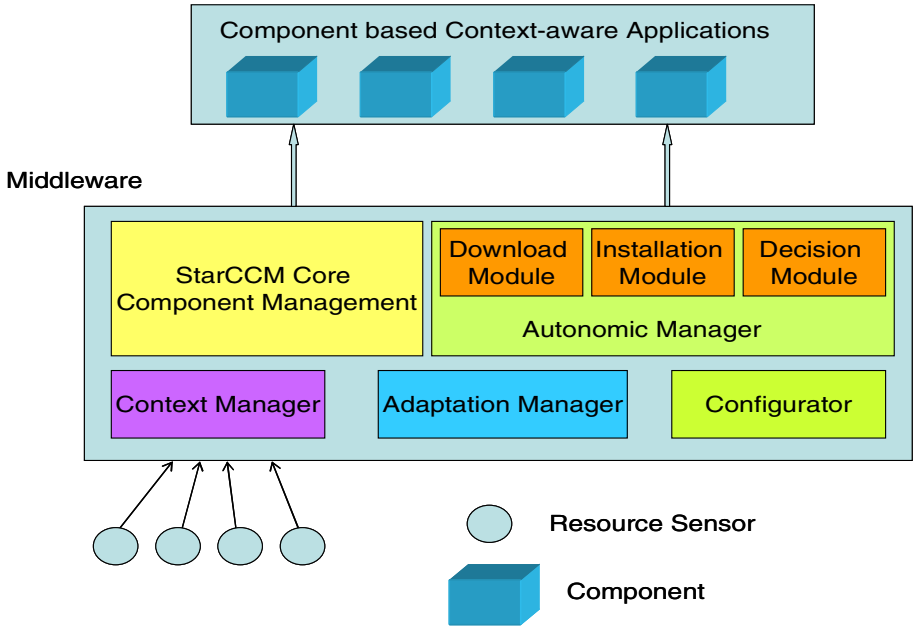


Fig. 2. Architecture of the Context-Aware Middleware

Adaptation Manager is responsible for reasoning on the impact of context changes on the application(s), and for planning and selecting the application variant or the device configuration that best fits the current context. As part of reasoning, the Adaptation Manager needs to assess the utility of these variants in the current context. The Adaptation Manager produces dynamically a model of the application variant that

best fits the context. We use the term "configuration template" to denote a model of an application variant where all variation points have been resolved.

Configurator is responsible for coordinating the initial instantiation of an application and the reconfiguration of an application or a device. When reconfiguring an application, the Configurator proceeds according to the configuration template for the variant selected by the Adaptation Manager. Thus, the Configurator carries out the adaptations decided by the Adaptation Manager by applying the configuration template. The Adaptation Manager and the Configurator are tightly coupled as they operate on a common information element: the configuration template.

Autonomic Manager provides the basis for realizing the dynamic, automatic binding of software components into concrete functionality as well as the dynamic replacement of a component with another. The components are downloaded and installed in the system; thereafter they realize their seamless plug-in to the system. Another issue lies in the capability to enable the dynamic replacement of one component with another during runtime. Specifically, the proposed system supplies the components with the appropriate characteristics that allow them to be properly initialized and self-configured in order to achieve on-the-fly replacement. The Download Module, which deals with the orchestration of the software transfer to the system, and other procedures, i.e. asserting the authenticity of the concerned component's source, and integrity checks. The Installation Module, which caters for post-download steps, as well as the installation and integration of the downloaded components in the system. The Decision Module, which defines certain actions and decisions for the configuration of the autonomic system, after evaluating its behavior. The latter may include QoS requirements, user preferences, business aspects as well as constraints concerning resources. Based on this information, this module deals with the selection of the most suitable software components to use in a given configuration of the autonomic system and triggers the downloading of missing components.

3 QoS-Aware Context Management

3.1 Architecture of the Context Management Infrastructure

As we have discussed above, with the development of the pervasive computing, dealing with the direct access to context information during the development of a context-aware[10,11,12] service or application is expensive, error-prone, and the applications will be complex and non-portable. So we must try to separate the context-aware infrastructure from the context-aware applications. The context-aware infrastructure deals with the activities like communicating with context sources, collecting context data, storing and managing context data, and finally transforming context data into higher level context and refine them according to the applications needs. All the actions are transparent to the end users and the applications supported may be more scalable. The middleware provides abstractions for the fusion of the sensor information to obtain high-level context information whereas the context-aware applications are responsible for adaptation and reaction to context changes. Furthermore, the middleware allows designers to build context-aware applications and to interact with context-aware services by providing a meta-model for describing

context and adaptation policies and the middleware communicates with the underlying execution environment to collect context information, processes them to identify relevant changes, and propagates those changes up to the context-aware components or adapt the applications to context changes.

As depicted in figure 3, the context manager provides two important interfaces to clients, the context listener and the context access. The clients can either request to be notified of certain events using the context listener interface (push), or they can use the context access interface to explicitly query context information (pull).

The context repository is the main entry point for clients to the context manager. The primary tasks of the context repository are to maintain a context model, register and notify listeners, give access to context elements, and keep registry of available components (sensors, reasoners and storage). In order to get access to a specific context element, a context client (such as the adaptation manager) either registers as an observer to that element, or directly accesses it via the context access interface.

The context sensors are components which provide context information to the context repository (a type of context source). Sensors can be wrappers around specialized hardware drivers, or legacy code used for monitoring context, such as battery, memory, and network information. This component is in charge of discovering, managing sensors and collecting raw data from them through agents. It shields the heterogeneity of all kinds of sensors and provides universal interfaces and transmits the raw context data. At the same time, it does some preprocess work, for example, it will filter the data by certain rules, wipe off redundancies and encapsulate them in a unified format. This component has to activate and deactivate agents according to the context to which applications or services are sensitive.

The context interpreter abstracts raw or low level context information into richer or higher level information according to interpretation rules described by using the

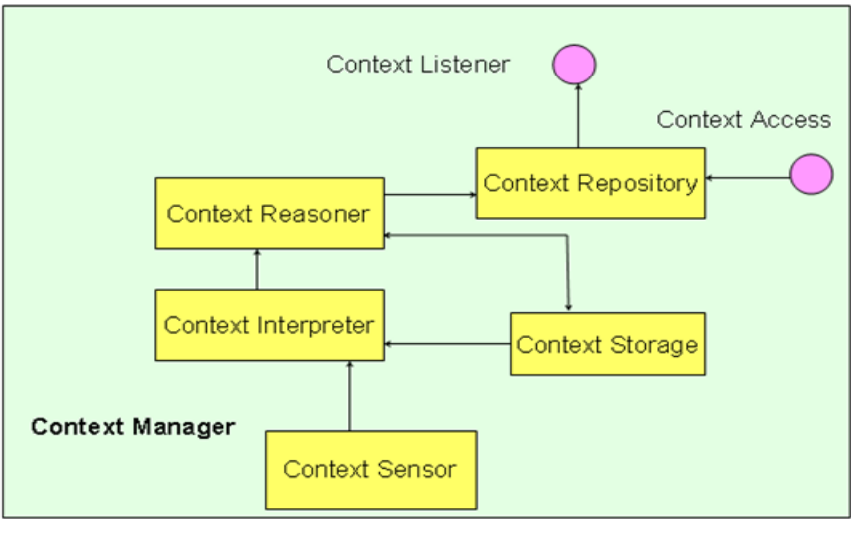


Fig. 3. Architecture of the Context Manager

context meta-model provided by the middleware. Furthermore, this component can fuse kinds of basic information into more comprehensive elements; for instance, it can merge temperature, luminance, humidity and other context within the room into a ROOM context.

The context reasoners can produce one or more context elements using other context elements as input. This component is used to filter context information to determine relevant ones, and notify the subscribed component of these context changes. It is an important task for the context manager to reduce context noise, by filtering out unnecessary context information, which is not relevant for adapting applications. The adaptation manager should only be notified when a significant change occurs in context. Consequently, the context reasoners need to implement filtering mechanisms. These mechanisms can vary from very simple, rule-based logic, to more advanced techniques. The reasoners are "plug and play" in order to make it possible to target reasoners according to different needs and domains. For example, the applications can provide the context manager with Quality of Context (QoC) requirements, such as precision and refresh rate.

The context storage keeps the track of historical context information which is often required in order to determine trends in context data (for example trends in user behavior, network stability, etc). The need for storage mechanisms was shown in Dey [9], where context widgets stored all context information they sensed.

3.2 Context Quality Measurements

In the recent years, context-aware computing has extracted a lot of attention from academic researchers and industrial practitioners. Context-aware systems usually make use of a large amount of sensed context information which is obtained from various physical sensors. Over the past decade, many context-aware applications have been built; however, few of them have been deployed in real life. One of the critical issues is quality of context; this issue is either ignored or not well addressed in the existing context-aware systems. The issues on quality of context may vary from type to type and from application to application; but it has two main factors. First, inconsistent contexts may often appear in context-aware systems because different sensors may produce different sensed data values which will lead to the inconsistency of sensor-based applications. Second, most sensors usually send sensed data to sinks periodically so that it is very difficult for computers to know what indeed happens in the time interval between two sensor signals. As a result, quality of context is always difficult to guarantee.

To measure quality of context, we propose three important parameters: Delay time, Context correctness probability, and Context consistency probability.

Delay Time. Delay time is the time interval between the time when the situation happens in real world and the time when the situation is recognized in computers. It is important to context-aware applications, because outdated contexts will not be useful to applications.

Context Correctness Probability. Due to the limitation of sensor technology, the accuracy of sensed data is difficult to guarantee. However, if we measure contexts through random sampling in a rather long period with recording the correct rate (the

probability that contexts in computers match situations in real world), we are able to provide quantity measurement for quality of context in a context-aware system.

Context Consistency Probability. Incorrect contexts often lead to context inconsistency. Context Consistency Probability measures the consistency rate of context information (the probability that contexts in computers are consistent), and it also could be obtained through long period random sampling.

A well-designed context-aware system should have low delay time, high context correctness probability and high context consistency probability. The three measurements have correlations with each other: outdated contexts with large delay time are usually incorrect and conflicting with current context information; and inconsistent contexts usually contain incorrect ones.

3.3 QoS-Aware Context Processing Procedure

We use logic inference to process contexts in our system. The detailed context processing procedure is shown in Figure 4.

The first step is the raw context gathering, in which raw contexts from various sensor sources are collected during a fixed short period. The second step is the inconsistency resolution. We resolve inconsistency among different raw contexts in this step because inconsistent raw contexts may lead to high-level inconsistent contexts that are more difficult to handle. We process raw contexts in a batch by batch manner instead of a piece by piece manner. Inconsistency in a batch of raw contexts

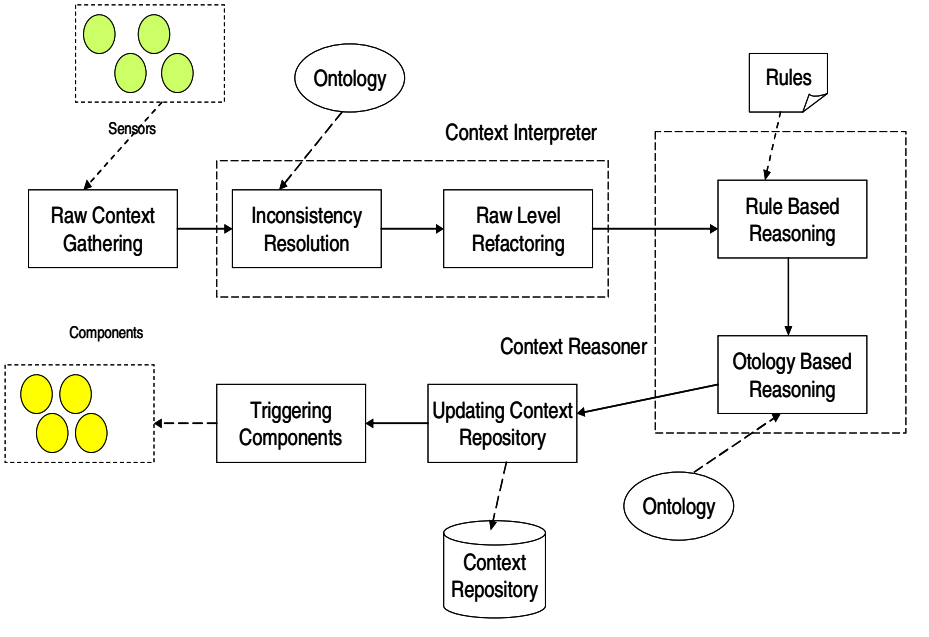


Fig. 4. Qos-aware Context Processing Procedure

should be cleaned prior to context reasoning so that the inconsistency of high-level contexts can be mitigated in certain degree. The third step is the raw level refactoring, in which we update the context repository with raw contexts, check the dependency graphs and refactor the ER graphs. Outdated or incorrect high-level contexts will be deleted in this step. If they are not removed, they will result in serious inconsistency among contexts after reasoning. Then, we apply rule-based reasoning to generate high-level contexts. The user-defined rules are of generic rules without negation and "or" operation. The two reasoners are configured as "traceable" in order to facilitate updating dependency graphs in context repository, though more memory is required. After that, we use inferred high-level contexts to update the context repository and notify components which register context triggers.

4 Policy Based Context-Aware Component Adaptations

As we have discussed before, the Autonomic Manager use the rules stored in the Adaptation Manager to complete the adaptations. A policy rule is defined as a rule governing the choices in behaviour of a managed system. Management action policies are defined as persistent, positive or negative, imperatives or authorities for a set of policy subjects to achieve goals or actions on a set of target objects. Informally, a policy rule can be regarded as an instruction or authority for a manager to execute actions on a managed target to achieve an objective or execute a change. An adaptation policy rule is usually made up of a trigger for the rule, which is often fired as a result of a monitoring operation, an action to perform in response to the trigger and a target for the action, which describes which managed part of the system to enforce the rule upon. Many policies will also contain some restrictions or guards confining the rule action to appropriate occasions.

Many traditional adaptable systems are composed of a single adaptation manager that is responsible for the entire adaptation process; i.e. monitoring, adaptation selection intelligence and performing the actual adaptation. Since the intelligence to select appropriate adaptations and the mechanism to perform these adaptations in embedded directly within the Adaptation Manager, this type of system becomes inflexible and inappropriate for general use.

By decoupling the adaptation mechanism from the Adaptation Manager, and removing the intelligence mechanism to select or trigger adaptation, the Adaptation Manager becomes more scalable and flexible. Since the user and the application are often most enabled to make informed choices, which are based on high-level contextual or semantic information about how a system should adapt, then it is logical that the user and the application help drive the adaptation of the system.

Policy specifications maintain a very clean separation of concerns between the adaptations available, the decision process that determines when these adaptations are performed and the adaptation mechanism itself. Policy specification documents are persistent text-based declarative representations of policy rules, where the document can usually be edited then interpreted to support the addition of new rules. Policy declaration files can be read, understood and generated by users, programmers and applications. In order for an adaptation to occur, the context changes that may trigger some adaptation must be monitored. The Context Manager should then leverage all

available context knowledge and intelligence to determine if some adaptation is required. A separate adaptation mechanism, controlled by the Adaptation Manager can then perform this triggered adaptation as a response to an adaptation request.

5 Conclusions

Ubiquitous computing allows application developers to build a large and complex distributed system that can transform physical spaces into computationally active and intelligent environments. Ubiquitous applications need a middleware that can detect and act upon any context changes created by the result of any interactions between users, applications, and surrounding computing environment for applications without users' interventions. The context-awareness has become the one of core technologies for application services in ubiquitous computing environment and been considered as the indispensable function for ubiquitous computing applications. The need for high quality context management is evident to the component-based middleware for it forms the basis of the component adaptation and the component deployment in the pervasive computing. Therefore, we suggest a holistic approach where context management is an integral part of a more comprehensive adaptation enabling middleware, thus enabling the development and support of context-aware, component-based applications. We have proved that such an approach is feasible, by developing and evaluating several different applications. Experimenting with the approach, we have identified ontologies and service oriented architectures as relevant approaches in relation to adaptation enabling middleware.

References

1. Weiser, M.: Some computer science problems in ubiquitous computing, Communications of the ACM (July 1993)
2. Roy, W., Trevor, P.: System Challenges for Ubiquitous & Pervasive Computing. In: Proceedings of the 27th International Conference on Software Engineering (May 2005)
3. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing Journal 5, 4–7 (2001)
4. Microsoft Corporation, An Introduction to Microsoft.NET, White Paper (2001)
5. Sun Microsystems. Entreprise JavaBeans Specification 2.0 (2002)
6. OMG CORBA Components Version 3.0, An adopted Specification of the Object Management Group (June 2002)
7. OMG. Specification for Deployment and Configuration of Component Based Distributed Applications (March 2003)
8. IST. COACH WP2: Specification of the Deployment and Configuration, D2.4 (July 2003)
9. Dey, A.: Providing Architectural Support for Building Context-Aware Applications, Ph.D. Thesis Dissertation, College of Computing, Georgia Tech (December 2000)
10. Dey, A., Abowd, G.D.: Towards a Better Understanding of Context and Context Awareness. Technical Report, GITGVU-99-22, Georgia Institute of Technology (1999)
11. Schmidt, A.: Ubiquitous Computing- Computing in Context, Ph.D. Thesis, Lancaster University, UK (2002)

12. Chen, G., Kotz, K.: A survey of context-aware mobile computing research. Department of Computer Science, Dartmouth College, Dartmouth, Technical report TR2000-381 (2000)
13. CORBA Components Version 3.0:An adopted Specification of the Object Management Group, OMG (June 2002)
14. Bruneton, T.C.E., Stefani, J.: The fractal component model (2004)
15. OMG, Specification for Deployment and Configuration of Component Based Distributed Applications (March 2003)
16. Information technology - open distributed computing - odp trading function, ISO/IEC JTC1/SC21.59 Draft, ITU-TS-SG 7 Q16 report (November 1993)
17. Mikalsen, M., Floch, J., Paspallis, N., Papadopoulos, G.A., Ruiz, P.A.: Putting Context in Context: The Role and Design of Context Management in a Mobility and Adaptation Enabling Middleware. In: MDM 2006 (2006)
18. Ayed, D., Taconet, C., Bernard, G., Berbers, Y.: An Adaptation Methodology for the Deployment of Mobile Component-based Applications. IEEE Computer Society Press, Los Alamitos (2006)

Design of High-Speed String Matching Based on Servos' Array

Wang Jie[1], Ji Zhen-zhou[2], and Hu Ming-zeng[3]

[1] Department of Computer Science and Technology,Harbin Institute of Technology, Harbin 150001, P.R. China
wj@pact518.hit.edu.cn
[2] Department of Computer Science and Technology,Harbin Institute of Technology, Harbin 150001, P.R. China
jzz@pact518.hit.edu.cn
[3] Department of Computer Science and Technology,Harbin Institute of Technology, Harbin 150001, P.R. China
mzh@hit.edu.cn

Abstract. String matching is a very important component of many network applications. Persistent increase of network bandwidth needs high performance string matching algorithms. Traditional software algorithms cannot fulfill requirement of content filter in high-speed network. Design of high-speed string matching based on servos' array in FPGA is presented by dynamic adjusting servos to obtain powerful parallel process performance. Through simulations and implemented on FPGA, feasibility and rationality are validated. With improving performance of automata algorithm, structure of storage and filter level, it can be improved further.

1 Introduction

The bandwidth of high-performance network interfaces has often exceeded the capabilities of workstations to process network data [1]. Content analysis of packets transmitting is emphasized more and more. So a good design of high-speed string matching is very important for high-speed network devices, such as gigabit routers, gigabit firewalls and so on, to reach approximate linear transmission with mirror delay especially. String matching is a very important component of many problems, including text editing, data retrieval, letter manipulation, WWW searching engine, computer virus characteristic codes matching, and data compression [2].

Algorithms for string matching have been researched and all kinds of improved measures have been employed to enhance matching performance or reduce complexity. Especially multi-string matching algorithms applied effectually for content fitter. Along with increase of bandwidth, traditional software method cannot adapt to high-speed network content filter. In a popular NIDS, such as Snort[3], 70% of total execution time and 80% of instructions are for string matching routines. So hardware methods for string matching have been focus on content filter.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 172–179, 2007.

2 Algorithms for String Matching

There are many algorithms for string matching that have been applied widely. These algorithms can be classified two kinds: algorithm based on jump table such as BM [4]. algorithm based on automata such as AC [5]. Some improved algorithms are also presented continuously.

These algorithms are usually implemented based on instructions rested with performance of CPU and operating system, but unfortunately it cannot generally achieve speed requirements. If implemented based on FPGA, it can be designed neatly according to characteristics of algorithms and suitable for high-speed network.

Related to jump table, automata is prone to implement based on FPGA. Automata will be decided if pattern aggregation is assured. Recently the research of multi-DFAs has been regarded as an effectual approach to implement high-speed string matching [6]. but it usually works on CAMs and SRAMs with higher cost.

With increase of size of Block RAMs in FPGA, hardware method of string matching and parallel strategies can easily implemented on FGPA with smaller extern SRAMs.

This paper provides a structure of servos' array on FPGA without CAMs which is suitable for high speed string matching. Initialization of automata will not be discussed because it can be produced by software programs and downloaded with initialization of FPGA.

3 Servos' Array on FPGA

A servo is made of an automata and a double data buffer. The size of automata is decided by number of states when data width if fixed. So the size of one servo is alterable with number of strings. But the system resource if fixed such as inner RAMs and gates of FPGA, we provide a structure of servos' array to furthest utilize fixed system resource.

3.1 Double Data Buffer

The function of data buffer for servos' array:

(1) Provide data stream for servos;
(2) Control buffer of input data and output data.

Fig.1 shows the nibble stream of data buffer (we suppose data width is 4-bit). Data buffer not only deals with input data and output, but also provide nibble stream for servos. But when nibble stream starts outputting, input data and output data for extern unit need stop. So the two functions work serially.

In graphics process, double data buffer is applied to avoid twinkle during screen switch. We also design double data buffer for servos' array: one is foreground and the other is background. Double data buffer can decrease or dispel data prepare time, and can also avoid buffer competition for different units.

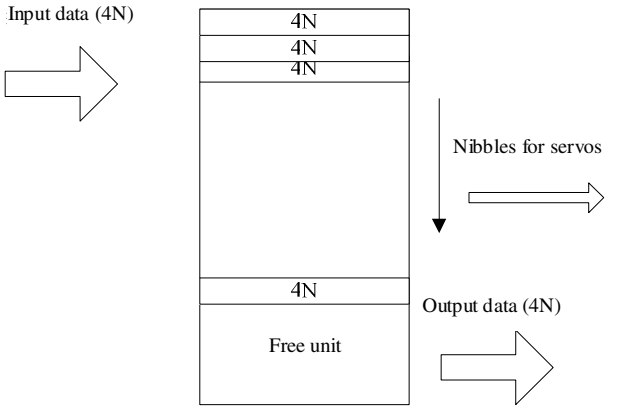


Fig. 1. The nibble stream of data buffer

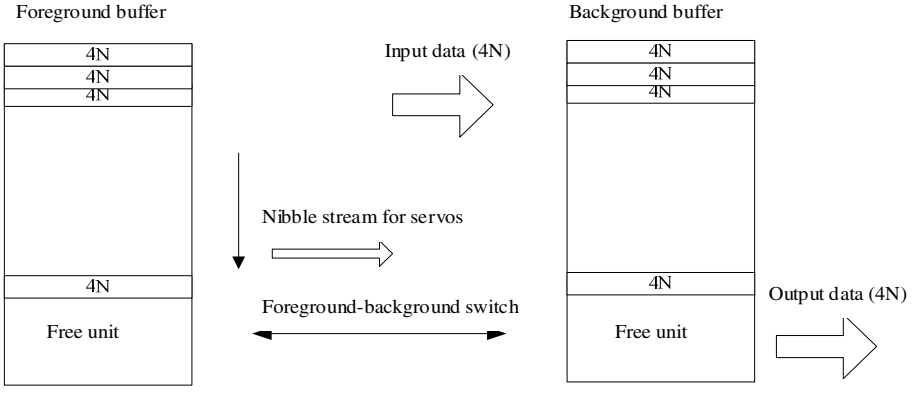


Fig. 2. The nibble stream of double data buffer

We denote that foreground buffer provides nibble stream and forbids data in-out, and background buffer deals with data in-out and prepares data. Generally foreground buffer is faster than background buffer. So when foreground buffer is done, buffer switch rapidly processes. Then foreground buffer changes background buffer and background buffer does contrary. Thus servos need not wait for nibble prepare, and eliminate effectually "data hunger" (servos wait nibble stream input). Double data buffer is showed as fig.2.

3.2 Servos' Array

Suppose the size of total resource is T, and the size of one servo is B, thus the number of servos is

$$n = \lfloor T/B \rfloor \tag{1}$$

All servos can work in parallel mode, so it can improve throughput greatly. Suppose the maximum of states of a servo is N_{max} ,we can always find a modern set which number of states is $N_{max}+1$, single servo cannot satisfy whether its size is. Thus we can combine appropriate servos to implement sting mach that is servos' array.

If number of states of a servo is $N_{standard}$ and resource of a servo is $R_{standard,}$ and total resource of system is $R_{total,}$ then maximum of servos is

$$n=\lfloor R_{total}/R_{standard} \rfloor \tag{2}$$

Suppose P_i is a modern set, and corresponding automata is S_0. If S_0 needs states N_p, the number of servos n_s is

$$n_s=\lceil N_p/N_{standard} \rceil \tag{3}$$

Due to formula (2) and (3), real number of parallel servos is

$$n_{total}=\lfloor n/n_s \rfloor \tag{4}$$

All servos need work in phase, and simply combine. We select one serve as master servo , and the others are slave servos. We can mark servos as 0, 1, … , n-1, and set a n-bit register MODE to tag a servos is master or slave. Suppose 1 means master and 0 means slave, the $MODE_i$ corresponding to P_i is showed as fig.3.

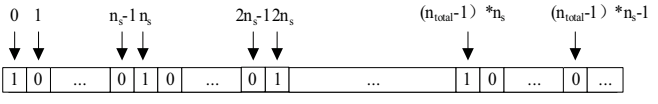


Fig. 3. Mode Register

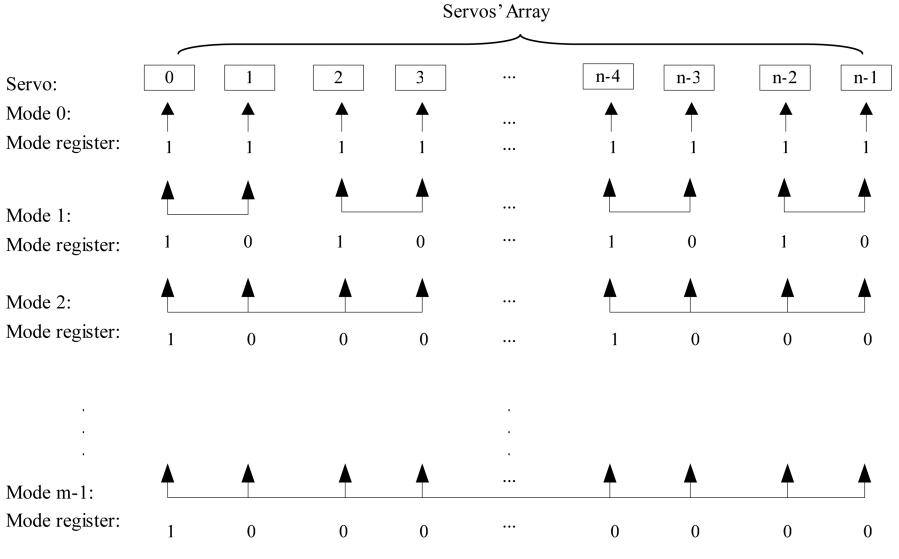


Fig. 4. Mode of Servos' Array

It can be find in fig.1 that it is a master servo every n_s servos. The value of mode register denotes a work mode, so the number of possible modes of n servos is

$$m=\lfloor \log_2 n \rfloor +1 \tag{5}$$

The higher parallel degree is, the smaller single servo is. The mode of servos' array can be showed as fig.4.

For example, we set 8 servos for servos' array, and suppose the maximum of state is 256, then all state of system is 256 * 8 = 2048. Encode a state in 11-bit, for string set P = { he, she, his, hers},the servos' array is configured as followed in fig.5(n_s = 1).

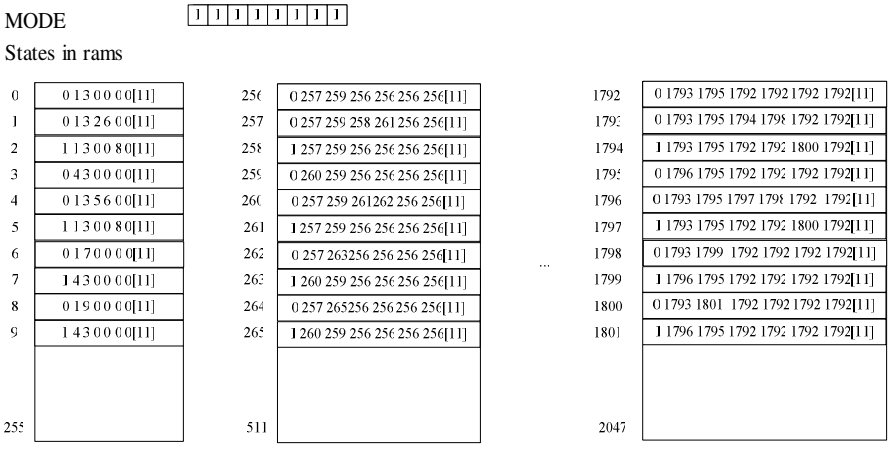


Fig. 5. Configuration of Servos' Array

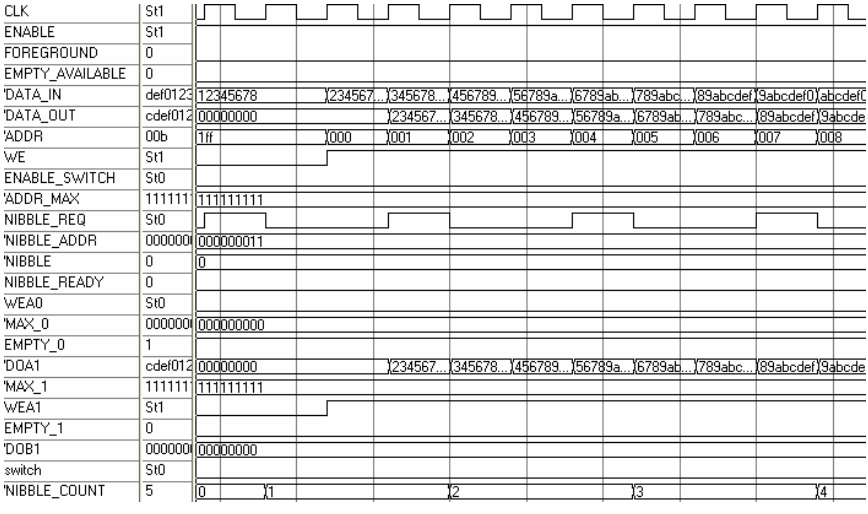


Fig. 6 Simulation Result of Writing to Background Buffer

4 Experiments

We simulated and implemented above scheme on Xilinx xc2v3000-4bg728.

Fig.6 shows the simulation result of writing to background buffer and fig.6 shows the simulation result of nibble stream from front buffer(by ModeiSim).

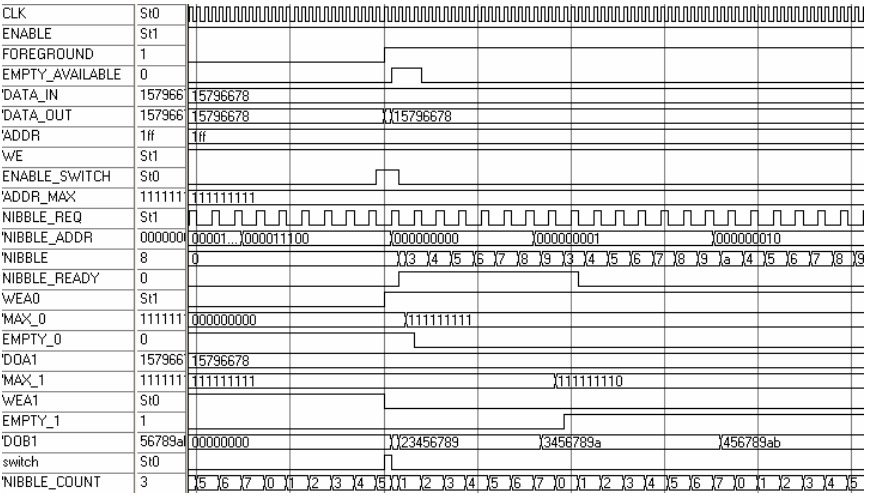


Fig. 7. Simulation Result of Nibble Stream from Foreground Buffer

Fig.8 shows the simulation result of writing state data into one servo of servos' array and fig.9 shows the simulation result of servo which is filtering.

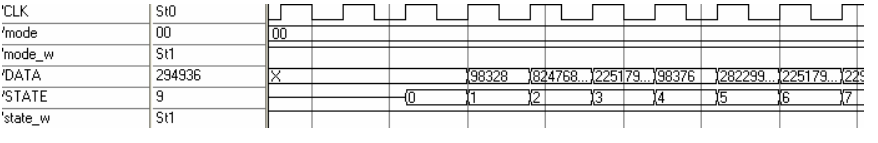


Fig. 8. Simulation Result of Writing State Data into Servos'Array

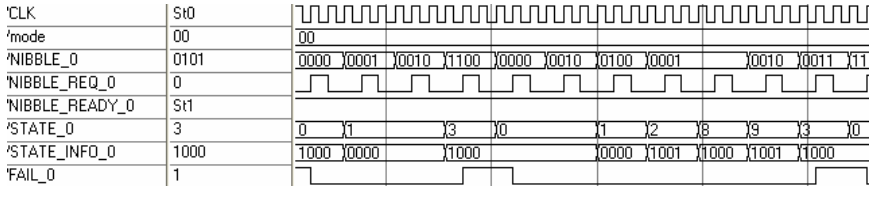


Fig. 9. Simulation Result of Servo which is Filtering

By above simulations, 8 servos can work successfully. Now correlative resource report of filter unit is given in table 1(by Synplify). This design can applied for giga-bit Ethernet data filter (string set is P). The worst time complexity intricacy is O(n), n is length of packet. Because initialization of automata is done with initialization of FPGA, we only need consider time of filter. If it is applied for gigabit stateful-inspect firewall, the max extra delay is 12 μ s, that is lower than national congeneric products(ms level).

Table 1. Resource Report of Filter Unit

Filter Part	Xc2v3000bg728-4
Filter | CLK-Estimated Frequency	141.3MHz
Filter | CLK-Requested Frequency	125.0MHz
Filter Register bit(Non I/O)	1559 (5%)
Filter Block Rams	20 of 96 (25%)
Filter Total Luts	1734 (6%)

5 Modifications for Gigabit Ethernet

Based above precept, we can enhance matching performance further to apply for gigabit Ethernet data filter.

5.1 Data Width

In chapter 3, we suppose data width is 4-bit that can decrease memory space. Along with improvement of data width, memory space will increase exponentially. If data width plus 1 bit when number of states is fixed, memory space will expand 4 times under perfect conditions.

Excellences of nibble input include:

(1) Higher throughput;
(2) Easily implemented only by inner block rams in FPGA;
(3) Suitable for MAC of 10/100M Ethernet.

In gigabit Ethernet, efficiency is lower by nibble input unless improve servos frequency. With high performance FPGA coming into the market, this method will be feasible. At the same time, high performance SRAMs such as ZBT Ram can work similarly with block rams that sustains bigger memory space.

5.2 Filter Level

Generally content filter works as independent unit. In stateful inspect firewall, content filter works after MAC process, packet filter and stateful inspect and so on. So for storage-transmit mechanism, content filter increases delays of packet transmission.

If filter level is on MAC process, no extra delays will be produced. When byte input for servos' array is enabled, content filter works with MAC process at the same time.

6 Summary

This paper presents a design of high-speed string matching based on servos' array in FGPA by dynamic adjusting servos to obtain powerful parallel process performance than software algorithms. Through simulations and implemented on FPGA, we validate feasibility and rationality.

Farther study will focus on improving performance of automata algorithm, structure of storage and filter level.

References

1. Bellows, P., Flid, J., Lehman, T.: GRIP: A Reconfigurable Architecture for Host-Based Gigabit-Rate Packet Processing. In: FCCM 2002 (2002)
2. Iyer, S., Awadallah, A., McKeown, N.: Analysis of a Packet Switchwith Memories Running Slower than the Line Rate. In: Proc. IEEE INFOCOM, pp. 529–537 (March 2000)
3. Antonatos, S., Anagnostakis, K.G., Markatos, E.P.: Generating Realistic Workloads for Network Intrusion Detection Systems. In: The Proc. ACM Workshop on Software and Performance, Redwood Shores, CA (2004)
4. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. Communications of the ACM 20, 762–772 (1977)
5. Aho, A.V., Corasick, M.J.: Efficient string matching: An aid to bibliographic search. Commun. ACM 18, 333–340 (1975)
6. Lu, H., Zheng, K., Liu, B.: A Memory-Efficient Parallel String Matching Architecture for High-Speed Intrusion Detection. IEEE Journal on Selected areas in Communications 24(10), 1793–1804 (2006)

An Efficient Construction of Node Disjoint Paths in OTIS Networks

Weidong Chen[1,2], Wenjun Xiao[1], and Behrooz Parhami[3]

[1] Department of Computer Science, South China University of Technology, Guangzhou 510641, China
[2] Department of Computer Science, South China Normal University, Guangzhou 510631, China
[3] Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560, USA
chwd2007@hotmail.com

Abstract. We investigate the problem of constructing the maximal number of node disjoint paths between two distinct nodes in Swapped/OTIS networks. A general construction of node disjoint paths in any OTIS network with a connected basis network is presented, which is independent of any construction of node disjoint paths in its basis network. This general construction is effective and efficient, which can obtain desirable node disjoint paths of length at most D+4 in $O(\Delta^2+\Delta f(N^{1/2}))$ time if the basis network of size n has a shortest routing algorithm of time complexity $O(f(n))$, where D, Δ and N are, respectively, the diameter, the degree and the size of the OTIS network. Further, for OTIS networks with maximally fault tolerant basis networks, we give an improved version of a conventional construction of node disjoint paths by incorporating the above general construction. Finally, we show the effectiveness and efficiency of these constructions applied to OTIS-Hypercubes.

1 Introduction

Optical transpose interconnection system (OTIS) networks are interesting interconnection networks for parallel computation and communication. An OTIS network with n^2 nodes is a two-level swapped architecture built of n copies of an n-node basis network that constitute its clusters. A simple rule for intercluster connectivity (node j in cluster i connected to node i in cluster j, for all $i \neq j$) leads to regularity, modularity, packageability, fault tolerance, and algorithmic efficiency of the resulting networks. The OTIS architecture has received considerable attention in recent years and has a special place among real-world architectures for parallel and distributed systems[1,11]. A number of algorithms have been developed for routing, selection/sorting[8,10], numerical analysis[5], matrix multiplication[14], and image processing[13].

Finding node disjoint paths or parallel paths in interconnection networks is one of the fundamental issues in design and implementation of parallel and distributed

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 180–189, 2007.

computing systems[4,16]. Parallel paths are useful in speeding up the transfer of large amounts of data between nodes and in providing alternative routes in cases of node or link failures [6]. From Menger's Theorem [15], there exist at least k parallel paths between any two distinct nodes in a network of connectivity k. In a general network, it is non-trivial to identify the parallel paths guaranteed by a given level of connectivity. For levels of connectivity greater than two, the identification of parallel paths is generally done using maximum flow algorithms which take $O(N^3)$ time, where N is the size of the network [16]. However, for the interconnection networks with special structures such as Hypercube networks, OTIS networks, and so on, flow techniques taking $O(N^3)$ time may be far from efficient.

Although some studies are related to general properties, including fault tolerance, of OTIS networks [3,9,17,18], so far all research work in this direction is only confined to OTIS networks with basis networks being maximally fault tolerant, and those proposed constructions of parallel paths in these OTIS networks are closely dependent upon the corresponding constructions in their basis networks [2,3,9].

In this paper, in a more general sense, we investigate the construction of the maximal number of parallel paths between two distinct nodes in any OTIS network whose basis network is connected. We propose an effective and efficient general construction of parallel paths in the OTIS network, which is independent of any construction of parallel paths in its basis network. This general construction can obtain desirable parallel paths of length at most D+4 in $O(\Delta^2+\Delta f(N^{1/2}))$ time if the basis network of size n has a shortest routing algorithm of time complexity $O(f(n))$, where D, Δ and N are, respectively, the diameter, the maximal node degree and the size of the OTIS network. Further, in the special case of a maximally fault tolerant basis network, we make an improvement over a conventional construction of parallel paths in such an OTIS network. Finally, the effectiveness and efficiency of these construction algorithms applied to OTIS-Hypercube are shown.

In the next section we describe OTIS networks. Section 3 presents the general construction of parallel paths in an OTIS network with a connected basis network. Section 4 gives the improved version of the conventional construction of parallel paths in those OTIS networks whose basis networks possess the maximally fault tolerant property. The application of these construction algorithms to OTIS-Hypercubes is discussed in Section 5. The conclusion is made in Section 6.

2 Preliminaries

Let G be a simple undirected graph (graph, for short) with vertex (node) set $V(G)$ and edge (link) set $E(G)$. For $v \in V(G)$, we denote by $deg_G(v)$ the degree of v in G, by $N_G(v) = \{u \in V \mid (v, u) \in E(G)\}$ the open neighborhood of v, and by $N_G[v] = N_G(v) \cup \{v\}$ its closed neighborhood. The maximum degree among the vertices of G is denoted by $\Delta(G)$ and the minimum degree by $\delta(G)$. The distance of between two nodes u and v, denoted by $d_G(u,v)$, is the length of a shortest path between u and v. The diameter D(G) of G is the maximal distance between any two nodes of G. Two paths from u to v are

node disjoint (also called parallel paths) if they have no common internal node. The connectivity of G is the minimal number of nodes in G whose removal can cause G disconnected or trivial. A graph G of connectivity $\delta(G)$ is maximally fault tolerant. Other notation and terminology used in this paper follow those in [15]. In the remainder of this paper, we use the terms graph and network interchangeably.

Definition 1. *OTIS* (*Swapped*) *network* [7,17]: *The OTIS* (*swapped*) *network* OTIS-Ω, *derived from the graph* Ω, *is a graph with vertex set* $V(\text{OTIS-}\Omega) = \{\langle g, p\rangle \mid g, p \in V(G)\}$ *and edge set* $E(\text{OTIS-}\Omega) = \{(\langle g, p_1\rangle, \langle g, p_2\rangle) \mid g \in V(G), (p_1, p_2) \in E(G)\} \cup \{(\langle g, p\rangle, \langle p, g\rangle) \mid g, p \in V(G)$ *and* $g \neq p\}$.

In OTIS-Ω, the graph Ω is called the basis (factor) graph or network. We refer to g as the cluster address of node $\langle g, p\rangle$ and p as its processor address. In an OTIS network, an intercluster (optical) link connects processor p of cluster g to processor g of cluster p for all $p \neq g$. No intercluster link is incident to processor g of cluster g. An example of OTIS networks is shown in Fig. 1.

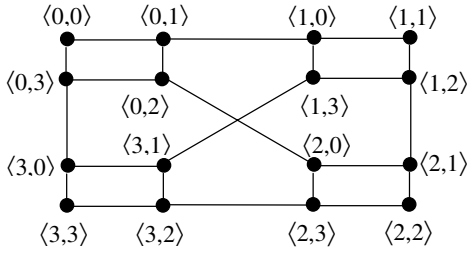


Fig. 1. An OTIS network with the basis graph C_4, a cycle of size 4

The following basis topological metrics of OTIS-Ω as functions of the corresponding metrics of Ω are derived from Definition 1 and similar expressions in [3,9]:

- $N=n^2$, where $N=|V(\text{OTIS-}\Omega)|$, $n=|V(\Omega)|$.
- $deg_{\text{OTIS-}\Omega}(\langle g, g\rangle)= deg_\Omega(g)$, and $deg_{\text{OTIS-}\Omega}(\langle g, p\rangle)=deg_\Omega(p)+1$ for $g\neq p$.
- $d_{\text{OTIS-}\Omega}(\langle g, p_1\rangle,\langle g, p_2\rangle)=d_\Omega(p_1, p_2)$, and for $g_1\neq g_2$, $d_{\text{OTIS-}\Omega}(\langle g_1, p_1\rangle,\langle g_2, p_2\rangle)=min\{d_\Omega(p_1, g_2)+d_\Omega(g_1, p_2)+1, d_\Omega(p_1, p_2)+d_\Omega(g_1, g_2)+2\}$.
- $\Delta(\text{OTIS-}\Omega)=\Delta(\Omega)+1$, and $\delta(\text{OTIS-}\Omega)=\delta(\Omega)$.
- $D(\text{OTIS-}\Omega)= 2D(\Omega)+1$.

The following results on parallel paths of an OTIS network have been given in [3].

Theorem 1. (*Day and Al-Ayyoub* [3]). *Let the graph* Ω *be connected.*

(1) *If* $g_1 \neq g_2$ *and* $deg_\Omega(p) = d$, *then there are d parallel paths between nodes* $\langle g_1, p\rangle$ *and* $\langle g_2, p\rangle$ *in* OTIS-Ω.

(2) *If* $\langle g_1, p_1\rangle$ *and* $\langle g_2, p_2\rangle$ *are two nodes in* OTIS-Ω *such that* $p_1 \neq p_2$ *and such that there are d parallel paths between* p_1 *and* p_2 *in* Ω, *then there are d parallel paths between* $\langle g_1, p_1\rangle$ *and* $\langle g_2, p_2\rangle$ *in* OTIS-Ω.

3 Constructing Parallel Paths in OTIS Networks with Connected Basis Graphs

In the section, we give an effective and efficient general algorithm for constructing parallel paths between two distinct nodes $\langle g_1, p_1\rangle$ and $\langle g_2, p_2\rangle$ in an OTIS network with a connected basis graph Ω.

3.1 Basis Idea

We first notice the following basis fact, which is easily derived from the rule for intercluster connectivity in OTIS networks: In cluster g_1 (g_2, respectively), every node of $N_\Omega[p_1]$ ($N_\Omega[p_2]$, respectively) is linked to one different cluster by an optical link if the node is not $\langle g_1, g_1\rangle$ ($\langle g_2, g_2\rangle$, respectively). Based on this fact, we construct parallel paths between $src=\langle g_1, p_1\rangle$ and $dst=\langle g_2, p_2\rangle$ as follows. Each of these paths begins with the source node *src*, immediately leaves cluster g_1 from a neighbor of *src* in cluster g_1 along an optical link, and then goes through successively at most two mediate clusters, until finally enters cluster g_2 at a neighbor of *dst* in cluster g_2 along an optical link prior to arriving at the destination node *dst*. If these mediate clusters are selected properly so that each mediate cluster can be passed through only by one of these paths, we will obtain desired parallel paths. See Fig. 2.

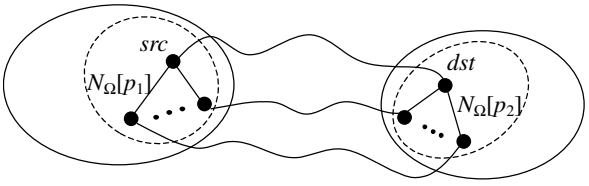


Fig. 2. An illustration of constructing parallel paths between *src* and *dst* in OTIS-Ω with Ω being connected for the case of $g_1 \neq g_2$ and $g_1 \notin N_\Omega[p_1]$ and $g_2 \notin N_\Omega[p_2]$

3.2 Algorithm

For the convenience of describing the algorithm, we need introduce some additional notations. We denote by $Path_\Omega(p, q)$ a shortest path from p to q in Ω, and by $\langle g, Path_\Omega(p, q)\rangle$ a shortest path from $\langle g, p\rangle$ to $\langle g, q\rangle$ in OTIS-Ω that is completely contained in cluster g. Let Y and Z be two disjoint subsets of $V(\Omega)$. A match M from Y to Z is a binary relation from Y to Z such that $|M|=min\{|Y|, |Z|\}$, and such that $(y, z)\neq(y', z')$ if and only if both $y\neq y'$ and $z\neq z'$ for all $(y, z), (y', z')\in M$. Obviously, a match M from Y to Z can be constructed in $O(\Delta^2(\Omega))$ time if Ω is represented by adjacency lists. In addition, we assume that a shortest routing algorithm of time complexity $O(f(n))$ in Ω is given, where n is the size of Ω.

Algorithm 1

Case I (g_1=g_2=g):

Step 1.1: Construct a path as follows based on a shortest path from p_1 to p_2 in Ω:

$\langle g, \text{Path}_\Omega(p_1, p_2)\rangle$, where $\text{Path}_\Omega(p_1, p_2)$=$p_1$→$\text{Path}_\Omega(y_0, z_0)$→$p_2$ for some $y_0 \in N_\Omega[p_1]$ and some $z_0 \in N_\Omega[p_2]$.

Step 1.2: Let S_0=$N_\Omega[p_1] \cap N_\Omega[p_2]$−$\{y_0, z_0\}$. For every $x \in S_0$, construct a path as follows: $\langle g, p_1\rangle$→$\langle g, x\rangle$→$\langle g, p_2\rangle$.

Step 1.3: Let S_1=$N_\Omega[p_1]$−S_0−$\{y_0, g\}$, S_2= $N_\Omega[p_2]$−S_0−$\{z_0, g\}$. Construct a match M between S_1 and S_2. Then, for each $(y, z) \in M$, construct a path as follows:

$\langle g, p_1\rangle$→$\langle g, y\rangle$→$\langle y, \text{Path}_\Omega(g, z)\rangle$→ $\langle z, \text{Path}_\Omega(y, g)\rangle$ →$\langle g, z\rangle$→$\langle g, p_2\rangle$.

Case II ($g_1 \neq g_2$):

Step 2.1: Construct a path as follows based on a shortest path from p_1 to g_2 and a shortest path from g_1 to p_2 in Ω: $\langle g_1, \text{Path}_\Omega(p_1, g_2)\rangle$→$\langle g_2, \text{Path}_\Omega(g_1, p_2)\rangle$, where $\text{Path}_\Omega(p_1, g_2)$=$p_1$→$\text{Path}_\Omega(y_0, g_2)$ for some $y_0 \in N_\Omega[p_1]$ and $\text{Path}_\Omega(g_1, p_2)$=$\text{Path}_\Omega(g_1, z_0)$→$p_2$ for some $z_0 \in N_\Omega[p_2]$.

Step 2.2: Let S_0=$N_\Omega[p_1] \cap N_\Omega[p_2]$−$\{g_1, g_2, y_0, z_0\}$. For every $x \in S_0$, construct a path as follows: $\langle g_1, p_1\rangle$→$\langle g_1, x\rangle$→$\langle x, \text{Path}_\Omega(g_1, g_2)$ →$\langle g_2, x\rangle$→$\langle g_2, p_2\rangle$.

Step 2.3: Let S_1=$N_\Omega[p_1]$−S_0−$\{g_1, y_0\}$ and S_2=$N_\Omega[p_2]$−S_0−$\{g_2, z_0\}$. Construct a match M between S_1 and S_2. Then, for each $(y, z) \in M$, construct a path as follows:

$\langle g_1, p_1\rangle$→$\langle g_1, y\rangle$→$\langle y, \text{Path}_\Omega(g_1, z)\rangle$→ $\langle z, \text{Path}_\Omega(y, g_2)\rangle$ →$\langle g_2, z\rangle$→$\langle g_2, p_2\rangle$.

3.3 Performance Analysis

The correctness of Algorithm 1 is stated in Theorem 2.

Theorem 2. *Let* Ω *be a connected graph,* $\langle g_1, p_1\rangle$ *and* $\langle g_2, p_2\rangle$ *be two distinct nodes in* OTIS-Ω. *Then*, *Algorithm* 1 *constructs at least d parallel paths between these two nodes in* OTIS-Ω, *where* $d = min\{deg_\Omega(p_1), deg_\Omega(p_2)\}$.

Proof. First, it is straightforward to check that the number of paths constructed by the algorithm is at least *d*. Secondly, in order to show all these paths are pairwise node disjoint, we note the following two facts: (i) no cluster, except for g_1 and g_2 (g in the case of g_1=g_2=g), is visited by more than one of these paths because S_0, S_1 and S_2 are disjoint sets, and (ii) all the segments of these paths contained in clusters g_1 and g_2 (g in the case of g_1=g_2=g) are pairwise node disjoint. In any case, the pairwise node disjoint property of the constructed paths is easily derived based on the aforementioned two facts, so the details of justifications are omitted. ■

Recall that $deg_{\text{OTIS-}\Omega}(\langle g, g\rangle)$=$deg_\Omega(g)$, and $deg_{\text{OTIS-}\Omega}(\langle g, p\rangle)$=$deg_\Omega(p)$+1 for $g \neq p$. From Theorem 2, we know that the number of parallel paths constructed by Algorithm 1 attains the maximum or less one than the maximum.

The performance of Algorithm 1 is given in the following theorem.

Theorem 3. *Let n and N be*, *respectively*, *the size of* Ω *and the size of* OTIS-Ω, Δ=Δ(OTIS-Ω), d=$d_{\text{OTIS-}\Omega}(\langle g_1, p_1\rangle, \langle g_2, p_2\rangle)$, *and l be the length of any path constructed by Algorithm* 1. *Then*,

(1) *the time complexity of Algorithm* 1 *is* $O(\Delta^2+\Delta f(N^{1/2}))$,

(2) $l \le D(\text{OTIS-}\Omega)+4$, *and*

(3) *if* $g_1 \ne g_2$ *and* $d=d_\Omega(p_1, g_2)+d_\Omega(g_1, p_2)+1$, *then* $d \le l \le d+6$ *holds for all the constructed paths, with at most* $|S_0|$ *exceptions in Step* 2.2.

Proof. (1) We first prove that the time complexity of the algorithm is $O(\Delta^2+\Delta f(N^{1/2}))$. On the one hand, generating the sets S_0, S_1 and S_2 requires $O(\Delta^2)$ time, and then obtaining the match M requires $O(\Delta^2)$ time. Based on these sets, on the other hand, constructing all required paths takes $O(\Delta f(n))$ time since at most Δ parallel paths need to be constructed. So, the total running time of the algorithm is $O(\Delta^2+\Delta f(n))$, namely, $O(\Delta^2+\Delta f(N^{1/2}))$ due to $N=n^2$.

(2) In the algorithm, obviously, any path contains at most two sub-paths like $Path_\Omega(g,p)$ for some $g, p \in V(\Omega)$, at most three optical links, and at most two other links (a link from $\langle g_1,p_1\rangle$ to its a neighbor in cluster g_1, a link from $\langle g_2,p_2\rangle$ to its a neighbor in cluster g_2). Considering $Path_\Omega(x,y) \le D(\Omega)$ for all x, $y \in V(\Omega)$, we have $l \le 2D(\Omega)+5 = D(\text{OTIS-}\Omega)+4$.

(3) In the case of $g_1 \ne g_2$, we consider the length l of any path constructed in Step 2.1 and Step 2.3. When $d= d_\Omega(p_1, g_2)+ d_\Omega(g_1, p_2)+1$, we have $l=d_\Omega(p_1, g_2)+ d_\Omega(g_1, p_2)+1=d$ for the path constructed in Step 2.1, and $l \le d_\Omega(y,g_2)+d_\Omega(g_1,z)+5 \le d_\Omega(p_1,g_2)+d_\Omega(g_1, p_2)+7=d+6$ for the path constructed in Step 2.3. Note that the last inequation is based on $y \in N_\Omega[p_1]$ and $z \in N_\Omega[p_2]$. Thus, we have $d \le l \le d+6$, as claimed. ■

Theorem 2 means there exist at least $\delta(\text{OTIS-}\Omega)$ parallel paths between any two distinct nodes in OTIS-Ω, since $\delta(\text{OTIS-}\Omega)=\delta(\Omega)$. From Menger's Theorem[15], we can derive that OTIS-Ω is maximally fault tolerant. From Theorem 3(2), moreover, we can obtain an upper bound of the fault diameter of OTIS-Ω, the diameter of the resulting graph from OTIS-Ω by removing at most $\delta(\text{OTIS-}\Omega)-1$ nodes.

Corollary 4. *Let* Ω *be a connected graph. Then,* OTIS-Ω *is maximally fault tolerant, and the fault diameter of* OTIS-Ω *is at most* $D(\text{OTIS-}\Omega)+4$.

4 Constructing Parallel Paths in OTIS Networks with Maximally Fault Tolerant Basis Graphs

In this section, we consider how to effectively and efficiently construct parallel paths in an OTIS network with a maximally fault tolerant basis graph.

4.1 A Conventional Algorithm

Provided that a parallel path construction method in the basis graph Ω is known, there is a straightforward construction of parallel paths in OTIS-Ω according to [3]. The idea of the construction in OTIS-Ω is as follows. If the source node and the destination node are in the same cluster, the construction of parallel paths between these two nodes is trivial since the construction of parallel paths in Ω is given. Otherwise, for each parallel path $\pi_i(p_1,p_2)$ from p_1 to p_2 in Ω, if the last 2rd node x_i on the path (namely, $x_i \in N_\Omega(p_2)$) such that $x_i \notin \{g_1,g_2\}$ then a path from $\langle g_1,p_1\rangle$ to $\langle g_2,p_2\rangle$ in OTIS-Ω is constructed as follows: $\langle g_1,\pi_i(p_1,x_i)\rangle \rightarrow \langle x_i,Path_\Omega(g_1,g_2)\rangle \rightarrow \langle g_2,x_i\rangle \rightarrow \langle g_2, p_2\rangle$, where $\pi_i(p_1, x_i)$ is a

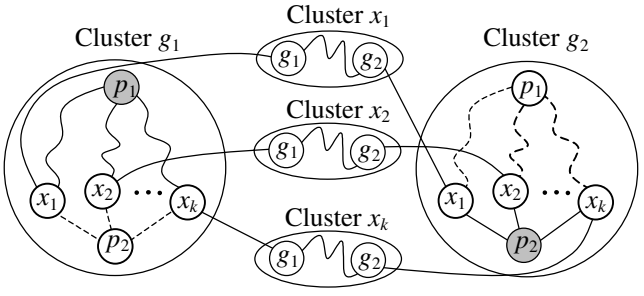


Fig. 3. An illustration of constructing parallel paths(shown solid) from $\langle g_1,p_1\rangle$ to $\langle g_2,p_2\rangle$ for the case of $g_1\neq g_2$ and $p_1\neq p_2$ and all $x_i\notin\{g_1,g_2\}$ in Algorithm 2

sub-path of $\pi_i(p_1,p_2)$. See Fig. 3. Notice that even if there exists i such that $x_i\in\{g_1,g_2\}$, one desired path can be constructed by cleverly using π_i as well as p_1 and/or p_2.

The construction described in [3] is here called Algorithm 2, which constructs k parallel paths between $\langle g_1, p_1\rangle$ and $\langle g_2, p_2\rangle$ in OTIS-Ω, where k is the number of parallel paths from p_1 to p_2 in Ω generated by the given construction of parallel paths in Ω. See [3] for the detailed description and proof of the correctness of the algorithm.

Now we give the performance of Algorithm 2. Assume the given construction of parallel paths in Ω requires $O(g(n))$ time for each path, and the given shortest routing algorithm requires $O(f(n))$ time, where n is the size of Ω. Let $\sigma+d_\Omega(p_1, p_2)$ be an upper bound of the length of any path between nodes p_1 and p_2 in Ω generated by the parallel path construction in Ω. The following theorem establishes the performance of Algorithm 2.

Theorem 5. *Let N be the size of* OTIS-Ω, $\Delta=\Delta$(OTIS-Ω), $d=d_{\text{OTIS-}\Omega}(\langle g_1,p_1\rangle,\langle g_2,p_2\rangle)$, *l be the length of any path constructed by Algorithm* 2. *We have*,

(1) *the time complexity of Algorithm* 2 *is* $O(\Delta g(N^{1/2})+\Delta f(N^{1/2}))$,

(2) $l\leq max\{D(\text{OTIS-}\Omega)+1+\sigma, D(\text{OTIS-}\Omega)+2\}$, *and*

(3) *if* $g_1=g_2$ *or*, $g_1\neq g_2$ *and* $d=d_\Omega(p_1, p_2)+d_\Omega(g_1, g_2)+2$, *then* $d\leq l\leq d+\sigma$ *holds for all the constructed paths*.

Proof. A proof of Theorem 5 is similar to the one of Theorem 3, and therefore is omitted. ■

4.2 An Improved Algorithm

From Theorem 3 and Theorem 5, we can see that there is no much difference between the performance of Algorithm 1 and one of Algorithm 2. However, we can improve Algorithm 2 in the length of constructed paths by combining it with Algorithm 1, so that the resulting algorithm can offer a guarantee that the length l of any constructed path is not much longer than the distance d between the source node and the destination node with a few possible exceptions for all the cases.

Recall that the distance d between two nodes $\langle g_1,p_1\rangle$ and $\langle g_2,p_2\rangle$ is $d_\Omega(p_1, p_2)$ for $g_1=g_2$, and $min\{d_\Omega(p_1, g_2)+d_\Omega(g_1, p_2)+1, d_\Omega(p_1, p_2)+d_\Omega(g_1, g_2)+2\}$ for $g_1\neq g_2$. The two items in the minimum function are independent from each other. From Theorem 3 and

5, we think Algorithm 1 is more effective than Algorithm 2 in the case of $g_1 \neq g_2$ and $d_\Omega(g_1, g_2)+d_\Omega(p_1, p_2)+2>d_\Omega(p_1,g_2)+d_\Omega(g_1,p_2)+1$, whereas Algorithm 2 is more effective than Algorithm 1 for the other cases. Therefore, Algorithm 2 can be improved by incorporating Algorithm 1, described as Algorithm 3.

Algorithm 3
If $g_1 \neq g_2$ and $d_\Omega(g_1, g_2)+ d_\Omega(p_1, p_2)+2>d_\Omega(p_1, g_2)+d_\Omega(g_1, p_2)+1$, then construct parallel paths by Algorithm 1; otherwise, construct parallel paths by Algorithm 2.

The performance comparison among the three algorithms is shown in Table 1. In Table 1, N is the size of OTIS-Ω, $D=D$(OTIS-Ω), l is the length of any constructed parallel paths, d is the distance between nodes $\langle g_1, p_1\rangle$ and $\langle g_2, p_2\rangle$, and $\sigma+d_\Omega(p_1, p_2)$ is an upper bound of the length of any path between nodes p_1 and p_2 in Ω generated by the given parallel path construction in Ω. In addition, Case *A* refers to the case of $g_1 \neq g_2$ and $d_\Omega(g_1, g_2)+ d_\Omega(p_1, p_2)+2> d_\Omega(p_1, g_2)+d_\Omega(g_1, p_2)+1$, and Case *B* to the other cases.

Table 1. Performance comparison among three algorithms

Alg.	Case *A*	Case *B*	Upper Bound on *l*	Time Complexity
Alg.1	$d \le l \le d+6$*	---	$D+4$	$O(\Delta^2+\Delta f(N^{1/2}))$
Alg.2	---	$d \le l \le d+\sigma$	$D+1+ max\{1,\sigma\}$	$O(\Delta g(N^{1/2})+\Delta f(N^{1/2}))$
Alg.3	$d \le l \le d+6$*	$d \le l \le d+\sigma$	$D+1+ max\{3,\sigma\}$	$O(\Delta g(N^{1/2})+\Delta f(N^{1/2}))$

Note. In Table 1, these two inequations with asterisk hold with at most $|S_0|$ exceptions, where $S_0=N_\Omega[p_1] \cap N_\Omega[p_2]-\{g_1,g_2,y_0,z_0\}$ in Step 2.2 of Algorithm 1.

5 An Example—Constructing Parallel Paths in OTIS-Hypercubes

In order to show the effectiveness and efficiency of these algorithms applied to an OTIS network with a specific basis network, in the section, we investigate these algorithms in the context of OTIS-Hypercubes, whose basis graphs are hypercube networks [12]. Hypercube networks and their many variants, including OTIS-Hypercubes, are popular graphs as the models of many interconnection networks. Some research works on OTIS-Hypercubes are reported [2,18]. We use Q_k to denote an k-dimensional hypercube network. Let n is the size of Q_k, namely, $n=2^k$ .It is known that Q_k has a shortest routing algorithm of time complexity $O(\log n)$. Moreover, it has been shown in [12] that Q_k is maximal fault tolerant, and there are k node-disjoint paths between any two nodes x and y in Q_k, each of which can be constructed in $O(\log n)$ time. Among these k paths, $d_{Qk}(x, y)$ paths are of optimal length $d_{Qk}(x, y)$ and k-$d_{Qk}(x, y)$ paths are of length $d_{Qk}(x, y)+2$.

From Theorem 3 and Theorem 5, the time complexity of each of these three algorithms applied to OTIS-Q_k is $O(\log^2 N)$, since $f(n)=O(\log n)$ and $g(n)=O(\log n)$ as well as $\Delta=\log n$. The performance comparison among these algorithms applied to OTIS-Q_k is given in Table 2, which straightforwardly comes from Table 1 due to $\sigma=2$. All notations in Table 2 are the same as ones in Table 1.

From Table 2, we can see that these algorithms are the same efficient in the context of OTIS-Q_k. However, Algorithm 3 slightly outperforms Algorithm 1 and Algorithm 2 with regard to the length of constructed parallel paths.

Table 2. Performance comparison among three algorithms for OTIS-Q_k

Alg.	Case *A*	Case *B*	Upper Bound on *l*	Time Complexity
Alg.1	$d \le l \le d+6$*	---	$D+4$	$O(\log^2 N)$
Alg.2	---	$d \le l \le d+2$	$D+3$	$O(\log^2 N)$
Alg.3	$d \le l \le d+6$*	$d \le l \le d+2$	$D+4$	$O(\log^2 N)$

6 Conclusion

In this paper, we have proposed an effective and efficient construction algorithm for the node-to-node disjoint path problem in an OTIS network with a connected basis network, which can find desired parallel paths of length at most $D+4$ in $O(\Delta^2+\Delta f(N^{1/2}))$ time if the basis network of size n has a shortest routing algorithm of time complexity $O(f(n))$, where D, Δ and N are, respectively, the diameter, the degree and the size of the OTIS network. Obviously, if the basis network with logarithmic degree has a shortest routing algorithm of logarithmic time complexity then the time complexity of the algorithm is $O(\log^2 N)$. The number of parallel paths constructed by the algorithm attains the maximum or less one than the maximum. In addition, in the special case of maximally fault tolerant basis networks, we make an improvement over a conventional construction of node disjoint paths in OTIS networks by incorporating the above algorithm. These obtained algorithms can replace a number of parallel path constructions in OTIS networks for specific basis networks. As an application of these algorithms to OTIS-Hypercubes, desirable node disjoint paths are obtained in $O(\log^2 N)$ time.

It is interesting to find efficient general algorithms for other disjoint path problems, such as node-to-set disjoint paths problem, set-to-set disjoint paths problem and *k*-pair nodes disjoint path problem, in OTIS networks.

Acknowledgement. Research of the first two authors was supported by the Natural Science Foundation of Guangdong Province, China(No.04020130).

References

1. Chatterjee, S., Pawlowski, S.: All Optical Networks. Comm. ACM 42(6), 74–83 (1999)
2. Day, K.: Optical Transpose k-ary n-cube Networks. J. Systems Architecture 50, 697–705 (2004)
3. Day, K., Al-Ayyoub, A.: Topological Properties of OTIS-Networks. IEEE Trans. Parallel and Distributed Systems 14(4), 359–366 (2002)
4. Hsieh, H.-J., Duh, D.-R.: Constructing Node-Disjoint Paths in Enhanced Pyramid Networks. In: Jesshope, C., Egan, C. (eds.) ACSAC 2006. LNCS, vol. 4186, pp. 380–386. Springer, Heidelberg (2006)

5. Jana, P.K.: Polynomial Interpolation and Polynomial Root Finding on OTIS-Mesh, Parallel Computing 32, 301–312 (2006)
6. Kim, J., Moh, S., Chung, I., Yu, C.: Robust Multipath Routing to Exploit Maximally Disjoint Paths for Wireless Ad Hoc Networks. In: Shen, H.T., Li, J., Li, M., Ni, J., Wang, W. (eds.) Advanced Web and Network Technologies, and Applications. LNCS, vol. 3842, pp. 306–309. Springer, Heidelberg (2006)
7. Marsden, G., Marchand, P., Harvey, P., Esener, S.: Optical Transpose Interconnection System Architecture. Optical Letters 18, 1083–1085 (1993)
8. Osterloh, A.: Sorting on the OTIS-Mesh. In: Proc. 14th Int'l Parallel and Distributed Processing Symp., pp. 269–274 (2000)
9. Parhami, B.: Swapped Interconnection Networks: Topological, Performance, and Robustness Attributes. J. Parallel and Distributed Computing 65, 1443–1452 (2005)
10. Rajasekaran, S., Sahni, S.: Randomized Routing, Selection, and Sorting on the OTIS-Mesh. IEEE Trans. Parallel and Distributed Systems 9(9), 833–840 (1998)
11. Rayn, J.: WDM: North American Development Trend. IEEE Comm. 32(2), 40–44 (1998)
12. Saad, Y., Schultz, M.: Topological properties of hypercubes. IEEE Transactions on Computers 37, 867–871 (1988)
13. Wang, C.-F., Sahni, S.: Image Processing on the OTIS-Mesh Optoelectronic Computer. IEEE Trans. Parallel and Distributed Systems 11(2), 97–109 (2000)
14. Wang, C.-F., Sahni, S.: Matrix Multiplication on the OTIS-Mesh Optoelectronic Computer. IEEE Trans. Computers 50(7), 635–646 (2001)
15. West, D.B.: Introduction to Graph Theory. Prentice-Hall, Englewood Cliffs, NJ (2001)
16. Wu, R.-Y., et al.: Node-disjoint paths in hierarchical hypercube networks. Information Sciences 177, 4200–4207 (2007)
17. Yeh, C.-H., Parhami, B.: Swapped Networks: Unifying the Architectures and Algorithms of a Wide Class of Hierarchical Parallel Processors. In: Proc. Int'l Conf. Parallel and Distributed Systems, pp. 230–237 (1996)
18. Zane, F., Marchand, P., Paturi, R., Esener, S.: Scalable Network Architectures Using the Optical Transpose Interconnection System (OTIS). J. Parallel and Distributed Computing 60(5), 521–538 (2000)

Pampoo: An Efficient Skip-Trie Based Query Processing Framework for P2P Systems*

Li Meifang[1], Zhu Hongkai[2], Shen Derong[1], Nie Tiezheng[1], Kou Yue[1], and Yu Ge[1]

[1] Department of Computer Science and Engineering,
Northeastern University, Shenyang, China, 110004
Li.Meifang@gmail.com, shenderong@ise.neu.edu.cn
[2] Baidu Inc., Beijing, China, 100080
zhuhongkai@baidu.com

Abstract. In this paper, we present Pampoo, a novel distributed framework for efficient query processing in P2P systems. We propose a new locality preserving data structure Skip-trie as its substrate. Skip-trie incorporates the advantages of skip graph with features of traditional trie. Thus, Pampoo can efficiently support various types of queries such as range queries and k nearest neighbor queries. We study the time cost of search and update operations on Skip-trie structure under our Pampoo framework. We further briefly present a repairing strategy to boost the robustness of Pampoo system. Extensive experiments are conducted to verify the effectiveness and efficiency of our approach.

1 Introduction

Distributed peer-to-peer (P2P) computing system has received growing attention for its wide area real-world applications in recent years, which brings forth the notion of sharing resources available at the edges of the Internet. The P2P paradigm specifies a fully distributed, self-organizing network design, where peers collectively form a system randomly. Therefore, it offers enormous potentials for extensive resource sharing, with remarkable features in terms of dynamics, scalability, resilience to failures, self-organizing and load balancing etc.. A large number of systems and architectures that utilize this technology have emerged since its initial success [1, 2, 3, 4].

Therefore, efficient query processing, as a key aspect in P2P systems, is increasingly important at present. Distributed Hash Table (DHT) has been a typical and the most widely applied strategy for peer routing, owing to its inherent characteristics such as scalability, load-balancing and fault-tolerance [1,2,5,6,7]. Nevertheless, DHT can only support exact queries since it adopts the cryptographic hash function such as SHA-1 to map application keys to their identifier space, which impairs the locality properties of the semantically close data items. Thus, DHT is marred by its deficiency in supporting range queries and other complex queries. Currently, several approaches have been proposed to remedy such shortcoming, such as Prefix Hash Tree[8,9], Skip Graph/Net[10,11,12], DP-tree[13] etc..

* Supported by the National Natural Science Foundation of China (60673139, 60473073, 60573090).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 190–198, 2007.

1.1 Motivations and Challenges

In our framework, Pampoo (means peer-bamboo in vigorous growth), we aim to design an efficient distributed data structure in support of a fairly rich set of possible data queries such as exact query for a key (e.g. a file name), partial query for a string (e.g. schema matching, prefix matching), k nearest-neighbor query for a numerical attribute, range query over various numerical attributes, multidimensional query, top-k query and point location query in Ad-hoc and sensor networks.

Applications of such queries include DNA databases, fuzzy systems, location-aware services, approximate searches for file names or data titles. Particularly, range query is significant in a large field of applications such as prefetching of web pages, enhanced browsing and efficient searching.

Therefore, in this paper, we mainly focus on a unified architecture in support of range query, i.e. locating resources whose keys lie within a certain specified range, which can also easily deal with the former three types of queries since they are the special cases for range query. For example, a prefix query for ISBN numbers in a book database acm.lib, we can resort it to range query constrained within the acm.lib range scope.
Our design of Pampoo intends to meet the following desired features:

1) Fault tolerance: the framework should adjust to the failure of some nodes, allowing simple repairing mechanism at small cost.
2) Efficient queries processing: the framework should support query processing in terms of the number of rounds of communication and number of messages that must be exchanged in order to complete requested query.
3) Small cost at network changes and data updates: the framework should flexibly tackle issues in node join/leave, data insertion/deletion as well as a necessary repairing.
4) Locality preserving: The structure should meet locality preserving in support of range queries that are based on an ordering of the data. This feature has certain practical advantages over DHT. For example, a search from c.neu.edu to k.neu.edu will not require contacting any node outside neu.edu, which not only reduce the searching scope, but also allow the message to be broadcast within neu.edu.

1.2 Contributions

The contributions of this paper are threefold:

- First, we propose a novel data structure Skip-trie which incorporates advantages of skip graph and the locality preserving feature of trie;
- Second, we present our Pampoo framework and study the time cost of search and update operations on Skip-trie structure;
- Third, we present a repairing strategy in support of the robustness and conduct extensive experiments to verify our approach.

The rest of this paper is organized as follows. We start by presenting our novel Skip-trie data structure in section 2; section 3 presents our Pampoo framework and study the operation cost under it. Extensive experiments are conducted in section 4. Section 5 describes a summary of related work, and finally section 6 draws the conclusion.

2 Skip-Trie Structure

2.1 Backgrounds

Trie, or prefix tree, is a common ordered tree data structure that is used to store an associative array of keys. The position in the tree shows what key a node is associated with. All the descendants of any one node have a common prefix of the string associated with that node, and the root is associated with the empty string. Though trie is commonly keyed by strings, it can also easily be adapted to serve similar functions of ordered lists of any construct, e.g., permutations on a list of digits, permutations on a list of shapes, etc.

Skip graph is a distributed data structure that extends the skip list into a distributed environment by adding redundant connectivity and multiple handles into the data structure [10,11]. On average, there are $O(log\ n)$ levels in skip graph. All keys appear in sorted order in the list at Level *0*. Each Level *i*, for $i > 0$, can now contain multiple linked-lists. Each key maintains a *membership vector*, which is a random string of bits. For each *i* greater than *0*, each node appears randomly in one of the many link lists in level *i* with two constraints. First, if node X is a singleton at level $i - 1$, it doesn't appear in any of the linked list at levels higher than $i - 1$. Second, for every linked list *L* at level *i*, there must be another linked list L' at level $i - 1$ where the elements in *L* are a subset of the elements in L'. Skip graph is highly concurrent and resistant to node failures. More importantly, skip graph does not employ a hashing function which allows it to support range queries, since logically similar keys will become neighbors in the skip graph. However, each key must store pointers to an average of two neighbors for each of the $O(log\ n)$ levels. The result is a cost of $O(log\ n)$ state *per key*. Besides, it is unclear how keys are assigned to machines in the system in skip graph, thus skip graph makes no guarantees about system wide load-balancing nor does it make any guarantees about the geographic locality of neighboring keys.

These two limitations of skip graph incur our interest in designing our own data structure skip-trie to address such problems in our designing of Pampoo framework.

2.2 Skip-Trie: Two-Layered Data Structure

For notational convenience, we assume the data items are (but not confined to) data base tuples of multi-attribute relations R, suppose the number of attributes in R is *n*, $R = \{A_1, A_2, ..., A_n\}$, with each attribute A_i ($1 \le i \le n$) being represented as a string (can be other constructs of ordered lists as well). Our Skip-trie is constructed under two steps. First, we dynamically build a reduced logical trie, aka., a longest prefix tree based on the strings(e.g. the attributes, the name ID of a peer in a physical network), which are mapped to trie. Evidently, there are no more than *n* leaf nodes in trie since some attributes may be the substring of others. We consider all the strings of a tuple *t* in R as a segment, which is uniquely identified by a primary key $key(t)$ that

should be an attribute A_i within R. Second, we hash the primary keys with an order preserving hash function, i.e., $h(key_i) = val_i$, and construct the skip graph based on those keys with values (non-redundant skip graph). A hash function is order preserving i.f.f. it satisfies the following property:

Given two input strings s_1 and s_2, $s_1 \prec s_2 \Rightarrow h(s_1) \prec h(s_2)$, where $\prec$ is the prefix operator.

Note here that originally, all the leaf nodes of the trie are assigned some values, but we only consider those keys with special interest (i.e. the primary keys). It is also worth noting that since no two primary keys can be identical, thus our skip graph layer is non-redundant, thus we coin it as NR-skip graph.

The fundamental idea of our approach is to make use of the skip graph for efficient routing, while trie for the locality preserving. The inherent features of both structures are capable of supporting range queries, which we have already addressed. The substantial number of pointers in merely skip graph approach makes it really hard to implement and maintain. Therefore, in our skip-trie structure, the NR-skip graph is constructed based only on the primary keys of the underlying trie structure. Given k primary keys in R, typically $k << n$, NR-skip graph will only maintain k nodes instead of n nodes, thus it is relatively non-densed and the complexity of our data structure is significantly reduced.

Inspired by the two-layer architecture in [14], we can also think of our skip-trie structure as being composed of two layers, with NR-skip graph as the upper layer and trie as the lower one, as is shown in Fig. 1.

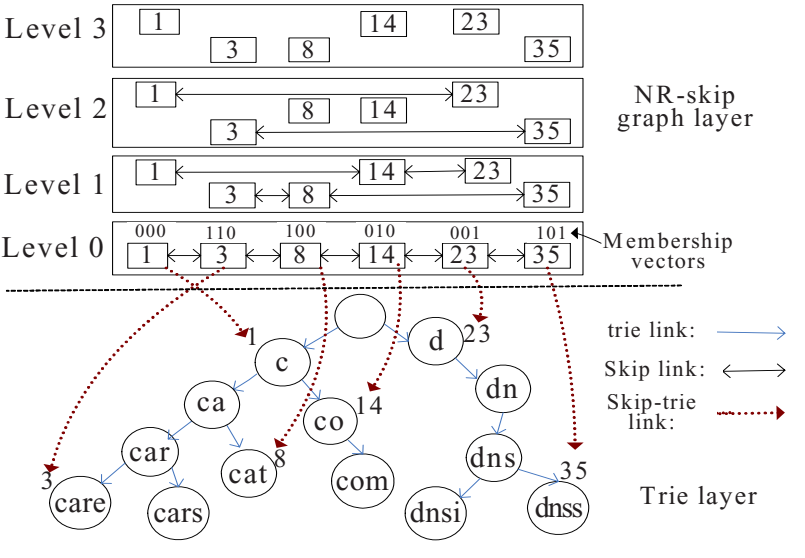


Fig. 1. Two-layered Skip-trie data structure

Observing the properties of skip graphs, we also have following theorem.

Theorem 1. In an NR-skip graph on k nodes, the height of every node is $O(\log k)$ with high probability.

Proof. It is identical to the theorem that with n nodes, the height of every node in skip graph is $O(\log k)$ with high probability.

3 Pampoo: A Skip-Trie Based Framework

In Pampoo, we denote the set of all the peers within the framework as PP , thus each peer $p_i \in PP$ is associated with a path $\vec{p}_i$ in the trie layer of Skip-trie, which corresponds to a binary string. The path may only involve the inner nodes of the trie, which is different from the trie-layered P-Grid architecture that each node only associates with the leaf node. Each peer stores $\vec{p}_i$ the prefixes of its path, thus allowing for efficient search routing.

Now we will discuss how the different operations are addressed in our Skip-trie based framework Pampoo.

3.1 Skip-Trie Search Algorithm

In skip graph, the search operation is achieved in a top-down manner. It is initiated by a top level node -skip layer seeking a key and proceeds down the lower level until it reaches level 0. However, in our skip-trie structure, we approach this operation quite differently. To search for a node with key from node X, we start from the trie layer first and proceed up to the NR-skip graph layer and then down to the trie layer again in a bottom-up-down manner, quite similar in family tree[15]. We incorporate the idea of shower algorithm in [14] that aims to process range queries concurrently, and propose our Skip-trie Search algorithm, which is illustrated as follows.

Algorithm 1. **Skip-trie Search Algorithm**: SSA(p_i, X)

1. Lookup $prefix\_path(p_i)$ // p_i caches the prefixes of its path
2. If $X \subseteq prefix\_path(p_i)$ Then
3. Return X
4. End if
5. $L\_p(X, p_i) \leftarrow$ Longest-prefix-search (X, p_i),
   $X' \leftarrow$ Trie-lookup $(X, prefix\_path(X))$ // find the closest nodes X' // to X with hashed value
6. $L\_p(X, p_i).key \leftarrow$ Map-trie-skip_Level_0 $(L\_p(X, p_i))$,
   $X'.key \leftarrow$ Map-trie-skip_Level_0 (X')
7. $X' \leftarrow$ Skip-level-search $(L\_p(X, p_i).key, X'.key)$
8. Return Trie-lookup (X, X')

In this algorithm, we start from the trie layer with the purpose of making use of the locality property and start the node near the destined node, thus perform the algorithm in an aggressively greedy way.

Lemma 2. The search operation in Skip-trie with n nodes in trie layer and k nodes in NR-skip graph layer takes $O(\log k + r)$ with high probability, where $r =$ maximal_cost{ Trie-lookup$(X, prefix_path(X))$, Trie-lookup(X, X') } .

Proof. In trie with n nodes, the lookup operation takes $O(n)$ amortized time cost, while in NP-skip layer, it takes $O(\log k)$ with high probability, thus verifies lemma 1. Moreover, our Skip-trie Search algorithm is processed in an aggressively greedy way, the cost of lookup operation in trie of n nodes is $O(\log n)$ with high probability (see [16] for details), thus practically the cost is much smaller than in Lemma with high probability.

3.2 Skip-Trie Update

We Address approaches of node join and node leave and their respective time cost in this section.

3.2.1 Node Join. Most of the work required to join (i.e., insert) a node is accomplished by calls to the search operation described in Section 3.1. When a new node X joins, we first find the node sharing the longest prefix with X, and then insert the suffix of X into the trie layer. If the string is a not a primary key, then the operation is over; however, if it is a primary key, we have further to do the insert operations in the NR-skip graph layer identical as described in skip graph.

Lemma 3. The insert operation in Skip-trie with n nodes in trie layer and k nodes in NR-skip graph layer takes $O(\log k + r)$ in expectation and $O(\log^2 k + r)$ with high probability, where r is the same as specified in section 3.1.

Proof. Similar to Lemma 2.

3.2.2 Node Leave. The algorithm for deleting a node X is straightforward. If X only belongs to the trie layer, we simply delete the path. If X also belongs to the NR-skip layer, then it is non-trivial. First, we have to enumerate the nodes with pointers to X and update them to the appropriate predecessor and successor. Then we delete the path in the trie layer.

Lemma 4. The delete operation in Skip-trie with n nodes in trie layer and k nodes in NR-skip graph layer takes $O(\log k + r)$ in expectation and $O(\log^2 k + r)$ with high probability, where r is the same as specified in section 3.1.

Proof. Identical to Lemma 3.

3.3 Repair Strategy

In this section, we describe a self-stabilization strategy in Pampoo that repairs our Skip-tree in case of node failures. If the node only lies in the trie layer, then we do not

have to take any measures since the system is not affected. However, if the node belongs to the NR-skip layer, we have to repair the system for robustness. Thus, the repair strategy mainly focuses on the NR-Skip layer: each node in NR-skip graph layer sends message to its neighbors periodically to see if they are alive. If one of the neighbors fails, then we try to fix the link to the next live neighbor. Our repair strategy works quite similar to that in Skip B-tree[17].

Since load is generally uniform in trie structure, our Skip-trie does not have to handle load balancing problem.

4 Experimental Evaluation

To evaluate the performance of our Skip-trie structure, we implemented Pampoo framework in Java and ran it over Planetlab [7], a testbed for large-scale distributed systems. In our implementation, each peer node is identified both physically by a pair of IP address and port number and logically by its position in the Skip-trie structure.

We compare Skip-trie with PHT with different distribution of data and range queries, since PHT also supports range queries and is easy to implement.

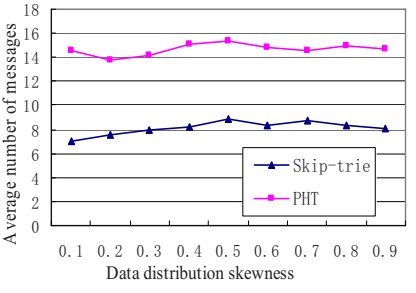


Fig. 2. Comparison of Skip-trie and PHT in number of message with different data distribution

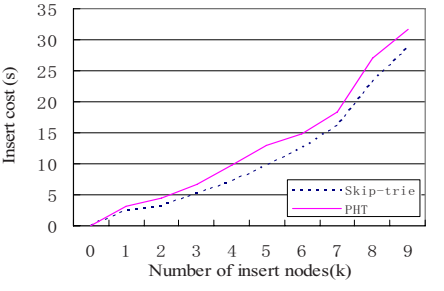


Fig. 3. Comparison of insert cost between Skip-trie and PHT

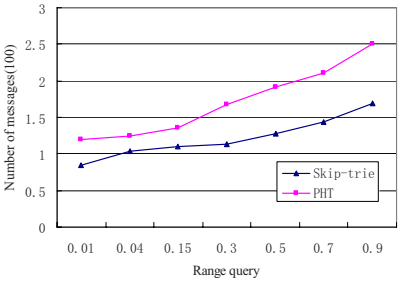


Fig. 4. Comparison of Skip-trie and PHT in number of message with different range query on uniform data distribution

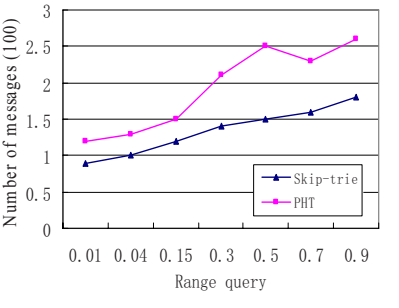


Fig. 5. Comparison of Skip-trie and PHT in number of message with different range query on skewed data distribution

From Fig. 2, we see that the number of messages in Skip-trie is much smaller that in PHT. Fig. 3 indicates that when the number of nodes to be inserted is small, we have fairly small insert cost; however, as the number increase, the time cost grows quickly.

Fig. 4 and Fig. 5 study the number of messages between Skip-trie and PHT under different data distribution. Skip-trie still performs much better than PHT and is not much affected by the skewness distribution.

5 Related Work

There are a wealth of work addressing issues in support of range query in P2P systems. To support approximate range queries, locality preserving hashing to hash ranges instead of keywords is used in [18]. An improvement of this approach to support exact range queries is proposed in [19]. The fundamental problem of these approaches is that the ranges themselves are hashed, and hence, simple key search operations are not supported or are highly inefficient.

Ganesan et. al. propose storage load balance algorithms combined with distributed routing structures which can support range queries [20]. Their solution may sup-port load balance in skewed data distributions, but it does not ensure balance in skewed query distributions. BATON is a balanced binary tree overlay network which can support range queries, and query load balancing by data migration between two, not necessarily adjacent, nodes[11]. In Mercury system, Bharambe et al support multi-attribute range queries and explicit load balancing, using random sampling[5]; nodes are grouped into routing hubs, each of which is responsible for various attributes.In terms of key search efficiency, support for range queries and storage load-balancing, there are some interesting novel structured overlay network abstractions which exhibit performance comparable to our trie-structured proposal: Skip Graphs [10, 11] which are based on skip lists [21].A detailed survey of search mechanisms in P2P systems, including range queries can be found in [22].

6 Conclusion

In this paper we propose a new two-layered data structure called Skip-trie which has several desirable properties. Skip-trie supports range queries in that it exploits the locality preserving feature in location of resources. Based on Skip-trie, we build a distributed P2P framework Pampoo, which aim to support efficient query processing and complex queries. We have studied the time cost of the basic operations in Skip-trie under our Pampoo framework and conducted extensive experiments to verify our approach.

Next, we will study the strategy to support top-k queries and multidimensional queries in our Pampoo framework.

References

1. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM 2001 (2001)
2. Druschel, P., Rowstron, A.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Middleware (2001)

3. Aberer, K., Punceva, M., Hauswirth, M., Schmidt, R.: Improving data access in P2P systems. IEEE Internet Computing 6(1), 58–67 (2002)
4. Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid: A Self organizing Structured P2P System. In: ACM SIGMOD Record (2003)
5. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware (2001)
6. Cuenca-Acuna, F.M., et al.: PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. Technical Report DCS-TR-487, Rutgers University (September 2002)
7. Ratnasamy, S., et al.: A scalable content-addressable network. In: SIGCOMM 2001 (2001)
8. Ramabhadran, S., Ratnasamy, S., Hellerstein, J., Shenker, S.: Brief Announcement: Prefix Hash Tree. In: Proc. of PODC 2004 (2004)
9. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM 2001, ACM Press, New York (2001)
10. Aspnes, J., Kirsch, J., Krishnamurthy, A.: Load balancing and locality in range-queriable data structures. In: ACM PODC 2004, ACM Press, New York (2004)
11. Aspnes, J., Shah, G.: Skip graphs. In: ACM-SIAM Symposium on Discrete Algorithms(January 2003)
12. Harvey, N., et al.: SkipNet: A scalable overlay network with practical locality preserving properties. In: Proc.of 4th USENIX Symp. on Internet Technologies and Systems (2003)
13. Mei Li.DP-tree: A Balanced Tree-based Indexing Framework for Peer-to-Peer Systems. In Proc. Of icnp 2006 (2006)
14. Datta, A., et,: al. Range queries in trie-structured overlays. In: Proc. of P2P 2005 (2005)
15. Zatloukal, K.C., Harvey, N.J.A.: Family Trees:An ordered dictionary with optimal congestion, locality, degree, and search time. In: 15th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 301–310. ACM Press, New York (2004)
16. Naor, M., Wieder, U.: Know thy neighbor's neighbor: Better routing in skip-graphs and small worlds. In: 3rd Int. Workshop on Peer-to-Peer Systems (2004)
17. Abraham, I., Aspnes, J., Yuan, J.: Skip B-Trees. In: Proc. of Opodis 2005 (2005)
18. Gupta, A., Agrawal, D., Abbadi, A.E.: Approximate Range Selection Queries in Peer-to-Peer Systems. In: CIDR 2003. 1st Biennial Conference on Innovative Data Systems Research (2003)
19. Sahin, O.D., Gupta, A., Agrawal, D., Abbadi., A.E., Peer-to-peer, A.: Framework for Caching Range Queries. In: 20th ICDE 2004 (2004)
20. Ganesan, P., Bawa, M., Garcia-Molina, H.: Online balancing of range-partitioned data with applications to peer-to-peer systems. In: Proc. of VLDB 2004 (2004)
21. Pugh, W.: Skip lists: A probabilistic alternative to balanced trees. Communications of the ACM 33(6) (1990)
22. Risson, J., Moors, T.: Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, Sydney, Australia (September 2004)

On the Implementation of Virtual Array Using Configuration Plane*

Yong-Sheng Yin [1], Li Li [2], Ming-Lun Gao[1,2], Gao-Ming Du[1], and Yu-Kun Song[1]

[1] Institute of VLSI Design, Hefei University of Technology, Hefei, Anhui 230009, China
YinYongSheng@hfut.edu.cn
[2] Intitute of VLSI Design, Nanjing University, Nanjing, Jiangsu 210093, China
{LiLi,GaoMingLun}@nju.edu.cn

Abstract. A new method of designing and using virtual array in pipeline reconfigurable system is presented. This method is based on the partition of the configuration data. Using this method not only is helpful to design the virtual hardware, but also is necessary to investigate the application algorithms oriented this virtual hardware. Basing on the analysis of the space-time graph and the configuration plane, this paper explores the structure and application of virtual array integrated in the MPRS (Multi-Pipeline Reconfigurable System), an in-house developed reconfigurable computing system that utilizes virtual pipeline. Finally, the design procedure of mapping the application to the virtual array and the programming procedure of using the MPRS are illustrated by examples. The experiment results show that the method is feasible and the performance of the MPRS with the virtual array nearly reaches the expected level.

1 Introduction

The fixed size of reconfigurable resource restricts the computing capability of reconfigurable system, which is one of the most important problems in reconfigurable computing. Based on this fact the concept of virtual hardware [1,2,3,4] have been presented, which means satisfying infinite resource requirement of algorithms by time division of finite hardware resource. Paper [1] surveyed a collection of important projects in this field. The virtualization of hardware is one of the basis objectives of studying dynamic reconfiguration.

In fact, similar restriction also exists in systolic array. The fact that one array can only be used to solve the applications under certain fixed size limits the application range of systolic array. Therefore some methods [5] including emulation method, partition method, LPGS (Local Parallel, Global Sequential) method, and LSGP (Local Sequential, Global Parallel) method have been proposed to resolve this problem.

* This work is supported by the National Natural Science Foundation of China under grant No. 90307011, 60373076.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 199 – 208, 2007.

The traditional LPGS/LSGP methods have been used as references when implement virtual hardware in reconfigurable systems, though some key processes must be changed to adapt the reconfigurable factors. Several projects show that the virtualization of hardware can be implemented by introducing *incremental reconfigurable* into the compute pipeline, designing buffers for intermediate data and creating corresponding control mechanism [4, 6]. In despite of some papers mentioned that the systems supporting virtual hardware have been completed, but most of them focused on describing the corresponding changes on the hardware and few of them explained what preparations of the target applications should be made for the virtualization of hardware, which is the key process when execute a algorithm using virtual arrays [3, 7].

On the other hand, there are some realistic difficulties existing in other projects to support the proposed method of designing virtual hardware. For instance, RaPiD implements large scale applications by storing multiple configuration data in local memory and then cyclically processing the compute data within the processing elements. This method makes the control logic in each processing unit too complex. More importantly, the data propagating between the neighbor units loses the inherent systolic rhythm because of the repeatedly unit-inside processing. All of these make the design of virtual array more difficult.

2 MPRS Architecture

We have implemented the MPRS that incorporates multiple 1-D arrays as coprocessor with a main processor. MPRS supports virtual array and the multiple 1-D arrays can optionally work in chained mode or parallel mode to explore the loop-level parallelism.

The structure of MPRS reconfigurable arrays is shown in Fig.1. The torus chain and the hierarchy buses are used as the interconnection backbone: torus chain connects the arrays and the buses connect the arrays with the storages. The first-level buses connect the main memory with the inner buffer. The second-level buses consist of the intra-array buses and the inter-array buses. The intra-array buses are used to connect the units in the same array with the buffer corresponding to that array, and the inter-array buses are used to connect the units in different arrays with buffer corresponding to that unit. In short, the hierarchy buses transport the input/output operation data, the reconfiguration data and the intermediate results.

In Fig. 1, the shadowed rPUs belong to one single linear array. S_FIFO represents the buffer used to store the intermediate result; D_FIFO represents the buffer used to store the operation data; C_FIFO represents the buffer used to store the reconfiguration data. The inter-array input buses make each unit have the capability of inputting data, but only the last unit of each array has the capability of outputting data.

Interconnecting neighbor rPUs (dashed arrows in Fig. 1) that belong to different arrays enhances the generality of MPRS to wider field of applications. In this way, the interconnection of MPRS extends from the torus chain to the torus mesh, which can utilize those mature algorithms based on 2-dimension mesh.The detail description about MPRS architecture and mapping method can be referred to the paper [8].

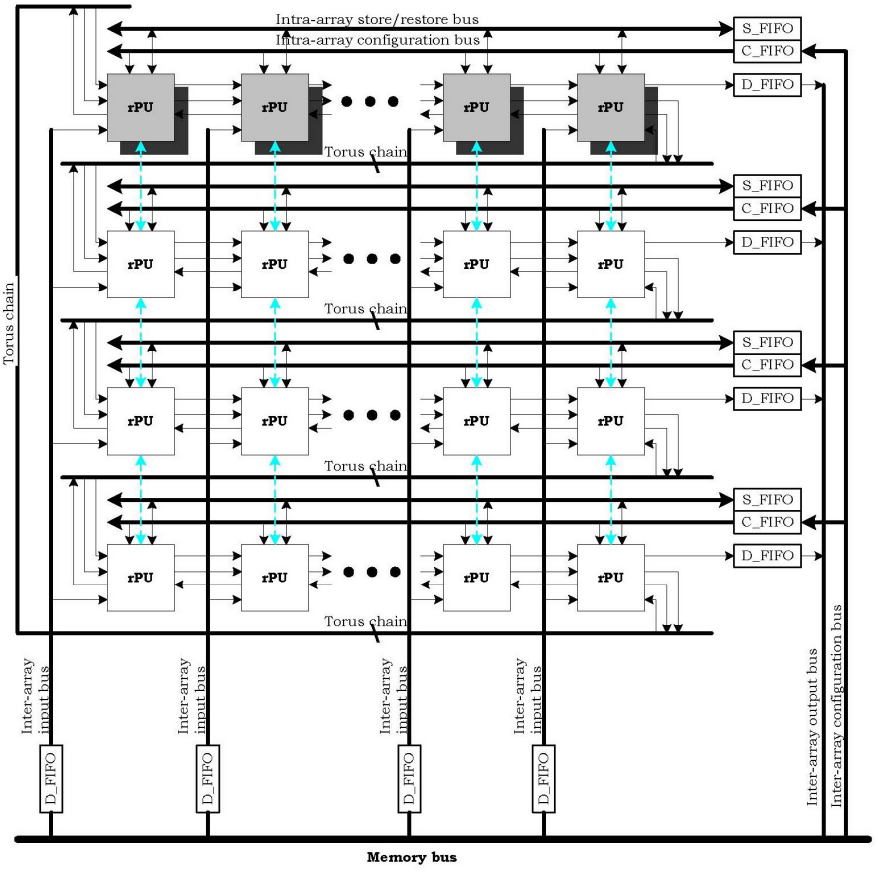


Fig. 1. MPRS architecture

3 Designing Virtual Array Using Configuration Plane

In the reconfigurable system utilizing virtual pipeline, the intermediate data between two sequent reconfiguring operations must be stored/restored in the right time, and the external data (including the configuration data and the computing data) for the pipeline must be arranged in correct order. In other word, we must predefine the organization and timing of these data.

The requirements on the various sequences of the input/output data should be met to compute various algorithms using pipeline array, and the same thing should happen to compute one same algorithm using different pipeline arrays. In fact, these requirements are decided by the specific configuring and executing of the physical array. The distinction of the data sequences is caused by two reasons: one is the data flow direction; the other is the data flow speed. The data flow direction can be transformed to fit MPRS anyway, so the data flow speed becomes the main factor need considering when design virtual arrays.

The data flow speed is corresponding with the number of pipeline registers in the rPU (reconfigurable Processing Unit). It is difficult to decide the data sequence when there are two or more registers in the Rpu's data path. The structures of MPRS array when compute matrix-vector multiplication and 1-D Convolution are shown in Fig.2 respectively. Only one pipeline register is used in the rPU of the former, while three registers (two is in the output data path and one is in the input data path) is used in the rPU of the latter. The design of virtual arrays for the latter will be more difficult than that for the former, because it is more difficult to figure out the organization and timing of input/output data, even more difficult to decide the time of storing/restoring the intermediate data.

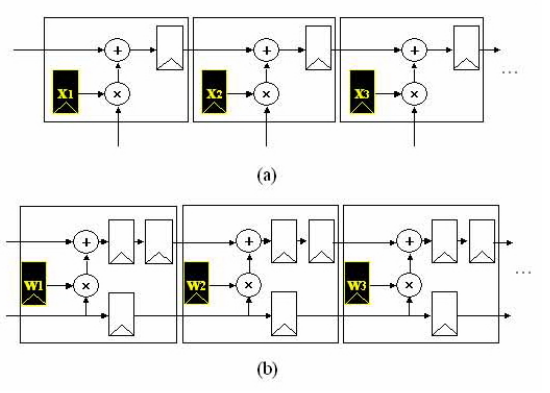


Fig. 2. Different data flow speed according to the different algorithms implemented on MPRS (a) matrix-vector multiplication; (b) 1-D convolution

The concept of "*configuration plane*" is proposed in this paper for those reasons mentioned above, and the corresponding design method based on the partition of the configuration plane is presented also. Using this method we can describe the configuration and execution of reconfigurable arrays directly, which make it easier to analyze the influence of the data flow speed on the virtual array, and to define the organization/timing of external data, and to decide the exact time of storing/restoring intermediate data. Only after all these key problems have been considered, can we design hardware structure correctly.

Suppose that an *n*-stage virtual pipeline is realized with an actual array including *m*-stage rPUs, as shown in Fig. 3. The whole array on the top of Fig. 3 is the virtual array: the real line shows the actual array, and the dashed line shows the virtual array simulated with the actual array. The space-time plane of the virtual array on the bottom of Fig. 3 shows the data operation of each rPU in each time step.

We partition these operations into several groups marked by a set of horizontal parallel lines in the time axis. The interval of these horizontal lines indicates the basic time step of the computing pipeline, and is called as "*virtual time slice*". All the operations in one same time slice should be done in one same time step if the actual array is large enough. However, if that actual array is smaller than the virtual array,

those operations must be processed in the different partitioned time steps. We hope make such partition clear. According to the maximal number k of the pipeline registers in one rPU, a set of directed biases whose slope is k can be drawn. All the operations along one same bias can be processed in the pipelined sequence, and then the final results can be achieved. We call this directed bias "*computing slice*". All the operations can be partitioned into many groups marked by the parallelograms along the computing slice, and those operations contained in one same parallelogram are the all operations needed when configure/execute the whole actual array one time. We call the parallelogram "*configuration plane*". It should be noted that only (m-1) operations on one configuration plane can be processed by the actual array simultaneously at the same time slice, and the remained one rPU is being configured at the same time. That is the cost that must pay out for the reconfiguration of array.

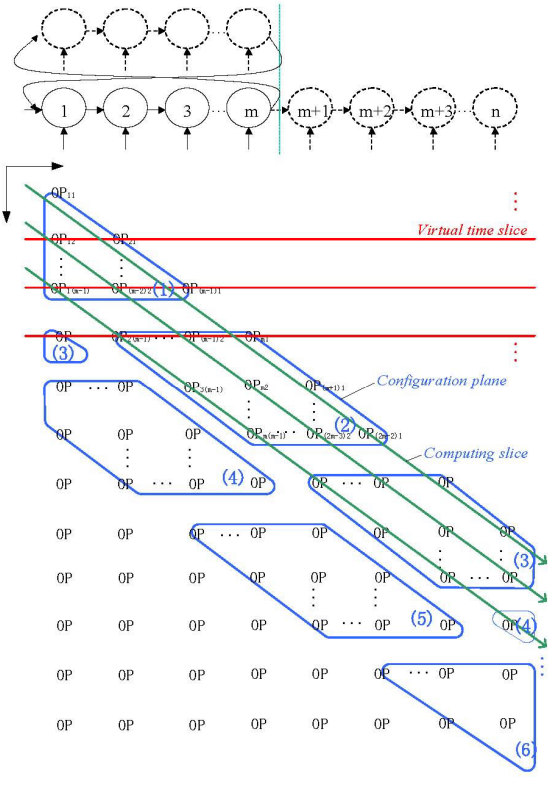


Fig. 3. Space-time plane of virtual array

The operation OP_{ij} of the rPU represents all possible operations that include inputting external data, executing arithmetic and logical operation (among the input data, local data and intermediate data) and outputting the results. A space-time plane of the actual array (Fig. 4.) can be achieved by arranging all these operations according to

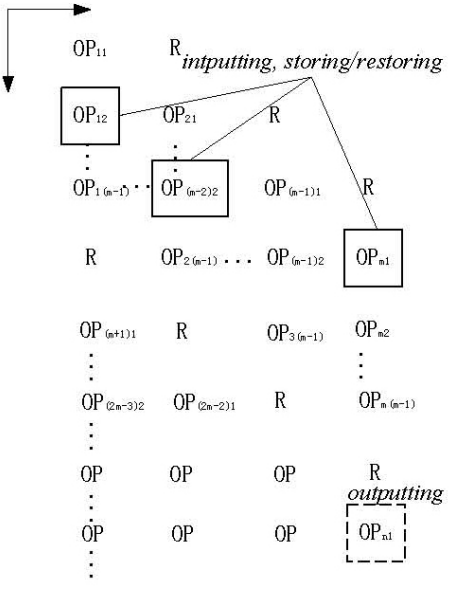


Fig. 4. Space-time plane of actual array

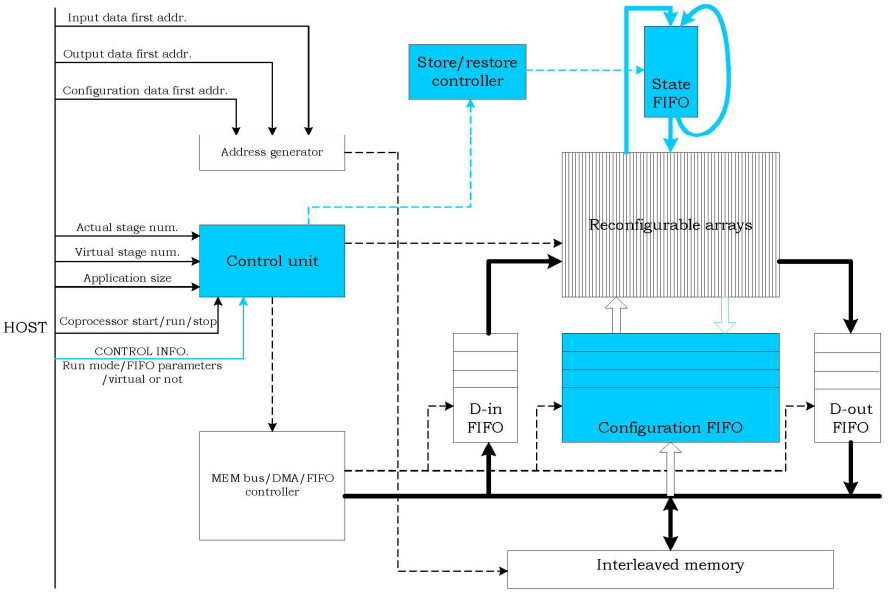


Fig. 5. Architecture of MPRS considering virtual array

the sequence number of configuration plane. From this new plane, we can decide when the operations of the same original virtual time slice are processed respectively; also can we decide the organization and timing of input/output data, and the storing/restoring time and contents of the intermediate data.

According to the analysis of configuration plane, we know that some special components should be designed in the MPRS if this system supports virtual array. All these components are marked with blue color in Fig. 5. The state FIFO is used to store the intermediate data and the store/restore controller decides when the store/restore operations start and finish. It should be noted that there are no additional storage devices for the storing/restoring of configuration data, and the configuration FIFO is reused for this purpose instead. As the precondition of this method, the reconfigurable array should work in a pure pipelined mode and the length of the virtual array should be smaller than the depth of the configuration FIFO. These two conditions can be met in our current MPRS implementation.

The computing time spend on processing the pipelined task using virtual array is determined by the scale of the actual array and the requirement of the task. Using the "configuration plane" method, it is easy to figure out (by the control unit of MPRS) the number of time slices needed to complete the given pipelined task. That number can be used to program the specific control register automatically, which is necessary to be definite for the designer of the system supporting virtual hardware.

4 Examples and Results

Matrix operation and motion estimation belong to the uniform linear recurrence applications fitting the MPRS array. Here we illustrate the design procedure and the programming procedure for the MPRS virtual array with these two examples and give the results finally.

4.1 Design and Programming Steps

The design steps of MPRS array list as follow, and the detail steps can be referred to another submitting paper "Mapping Algorithms to Multi-Pipeline Reconfigurable System" for limited space.

(1) Design the serial algorithm;

(2) Design the single assignment program by extending the index of the input variable;

(3) Construct the DG (Dependency Graph) according to the extended space-time index;

(4) Draw the DGRV (DG with Reconfigurable Variable) by localization and reconfigurablization processes;

(5) Design SFG (Signal Flow Graph) through projecting and scheduling the DGRV;

(6) Mapping the SFG into the multiple MPRS arrays simultaneously when the MPRS works in the parallel mode or mapping the SFG to the single chained MPRS

array when works in the chained mode. Working in the virtual mode is transparent for the mapping process;

(7) Reflect the mapping results into the different fields of the configuration word that will be used to reconfigure the MPRS arrays dynamically when perform computing.

The programming steps of MPRS list as follow.

(1) Prepare the configuration data achieved from above processes.

(2) Prepare the computing data achieved from the target application.

(3) Create the configuration/computing data file that will used by the main program. Firstly, draw the space-time plane of virtual array (like Fig. 3) and the space-time plane of actual array (like Fig. 4) respectively. Secondly, arrange the configuration data in proper sequence according to the configuration plane and organize the computing data properly according to the space-time plane of actual array. Finally, create the data files.

(4) Program the specific registers in MPRS to provide enough information for the system to run properly. This is done by creating main program.

(5) Compile the main program and run the executable code on the MPRS.

4.2 Matrix-Vector Multiplication

After implementing the procedures mentioned above, all the needed data and program are ready for running on the MPRS that works in the virtual mode.

Suppose that x is a 8-dimention vector, A is a $m\times8$ matrix, and $y=Ax$ is a m-dimension vector. Here, m represents the scale of matrix-vector multiplication, and its value changes from 32 to 4096.

In this experiment, various scales of array are used to compare the efficiency of the virtual array with that of the normal array. In the first situation, the array consists of 4 rPUs; in the second situation, the array consists of 8 rPUs. The execution periods of SimpleScalar [9] and MPRS on different application scales are shown in Fig. 6(a). In this figure, the horizontal axis represents the scale of matrix-vector multiplication, and the vertical axis represents the number of execution periods. MPRS_A shows the performance of MPRS using normal array, and MPRS_V shows the performance using virtual array. It can be seen that the number of periods spent by MPRS is much smaller than SimpleScalar. Fig. 6(b) shows the speedup factor of MPRS vs. SimpleScalar. It can be seen that the speedup factor when using virtual array is nearly equal to that when using normal array.

We can draw another conclusion by analyzing the periods spending respectively on the MPRS array and on the whole MPRS system. The system overhead should occupy the larger proportion of the total execution periods if the scale of applications is not large enough. In this situation, the number of rPU in the array can be effectively reduced by using virtual array. Inversely, when the application scale is large enough and the memory access time is short enough, the expense of speed caused by using virtual array should be considered fully.

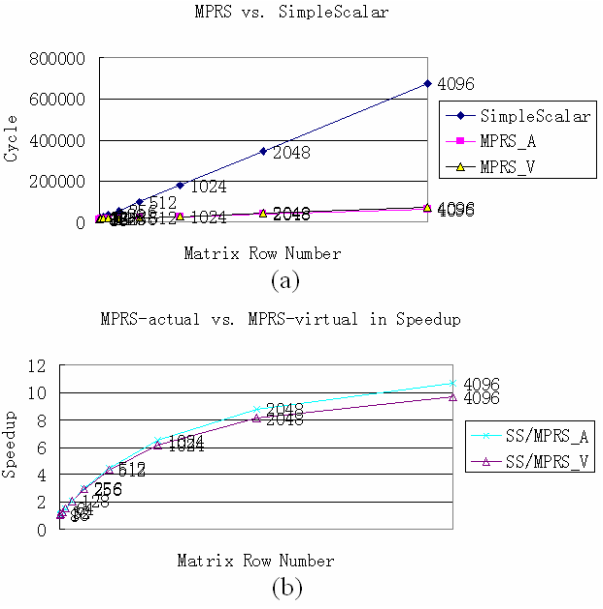


Fig. 6. (a) The number of execution periods of SimpleScalar and MPRS; (b) speedup factor

4.3 Motion Estimation

Since motion evaluation (ME) occupies the 98% processing time in video compressing and 42% in decompressing [10], it is important to enhance the execution speed of ME. One of most popular ME algorithm is the Full Search Block Matching (FSBM). FSBM can be expressed as follows:

$$MAD(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} | R(i,j) - S(i+m, j+n) | \qquad -q \le m,n \le q$$

In the parallel work mode, MPRS can complete one FSBM of the standard MPEG scale (with N＝8, q＝8). Fig. 7 also shows the different results in other systems, in

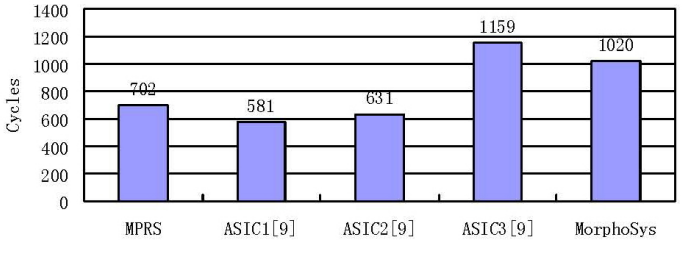


Fig. 7. Execution cycles of motion evaluation

which the ASICs have the special optimization for the FSBM [11]. Moreover, the same application in Pentium MMX needs 29000 cycles. It can be concluded that the speed of our MPRS in ME execution is 10 times faster than the general-purpose processor, also faster than the MorphoSys who is the similar reconfigurable computing system, and near the ASIC products.

5 Conclusions

This paper proposes a method of designing virtual hardware and exploiting target algorithms on it. The correct experiment results demonstrate that the method based on "configuration plane" is feasible. This new method can be applied to our MPRS system as well as to other systems using incremental reconfiguration.

References

1. Plessl, C., Platzner, M.: Virtualization of Hardware - Introduction and Survey. In: Proceedings of the International Conference on ERSA, pp. 63–69 (2004)
2. Hauck, S., Fry, T.W., Hosler, M.M., Kao, J.P.: The Chimaera Reconfigurable Functional Unit. IEEE Trans. on VLSI Systems. 12, 206–217 (2004)
3. Cronquist, D.C., Fisher, C., Figueroa, M., Franklin, P., Ebeling, C.: Architecture Design of Reconfigurable Pipelined Datapaths. In: Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI, pp. 23–40 (1999)
4. Cadambi, S., Weener, J., Goldstein, S.C., Schmit, H., Donald, E.: Managing Pipeline-Reconfigurable FPGAs. In: ACM/SIGDA International Symposium on FPGAs, pp. 55–64 (1998)
5. Lorenzelli, F., Yao, K.: Integral Matrix-Based Technique for Systematic Systolic Design Integration. The VLSI Journal 20, 269–285 (1996)
6. Schmit, H., Cadambi, S., Moe, M., Goldstein, S.C.: Pipeline Reconfigurable FPGAs. The Journal of VLSI Signal Processing 24, 129–146 (2000)
7. Goldstein, S.C, Schmit, H., Budiu, M., Cadambi, S., Moe, M., Taylor, R.R.: PipeRench: A Reconfigurable Architecture and Compiler. IEEE Computer 33, 70–77 (2000)
8. Yin, Y.S., Li, L., Gao, M.L.: The Reconfigurable System Based on Multi-Pipeline (in Chinese). Microelectronics & computer 10, 88–91 (2005)
9. Burger, D., Austin, T.M.: The SimpelScalar Tool Set, version 2.0. University of Wisconsin-Madison Computer Sciences Department Technical Report #1342 (1997)
10. Miyamori, T., Olukotun, K.: REMARC: Reconfigurable Multimedia Array Coprocessor. IEICE Trans. on Inf. and Syst. E82–D, 389–397 (1999)
11. Singh, H., Ming-Hau, L., Lu, G., Kurdahi, F.J., Bagherzadeh, N., Chaves Filho, E.M.: MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications. IEEE Trans. on Computers. 49, 465–481 (2000)

Analysis on Memory-Space-Memory Clos Packet Switching Network

Xiangjie Ma, Yuxiang Hu, Junpeng Mao, Julong Lan, Lian Guan, and Baisheng Zhang

Information Engineering Institute, PLA Information Engineering University
National Digital Switching System Engineering & Technological Research Center
Zhenzhou, Henan, 450002, P. R. China
maxiangjie100@163.com,
{mxj,hyx,mjp,ljl,gla,zbs}@mail.ndsc.com.cn

Abstract. Memory-Space-Memory (MSM) Clos packet switching networks are the next step in scaling current crossbar switches to many hundreds or few thousands of ports. Clos networks had been studied and applied quite well in circuit switching system, with much attentions paid to its non-blocking property to decrease call blocking rates. In contrast, for packet switching systems, more care is taken to per-packet based forwarding performance of the switching networks. MSM Clos network has the merit of keeping packet sequence and therefore is quiet adapt to packet switching fabric. By way of buffering architecture, MSM Clos network is quite similar to the CIOQ Crossbar based single stage switching fabric, which promotes us to extend the results of CIOQ matching an OQ switch [1] to MSM Clos networks. Meanwhile, although the CIOQ switch can emulate an OQ switch, it needs cell insertion algorithm and stable matching algorithm with high information complexity and computing complexity. This has prevented its application seriously in new generation of routers with high speed linking rates and large port numbers. So we propose a new method of Per-Input OQ Emulation (PIOE), including both new cell insertion and scheduling algorithm (PVPP-CIP and -CSP) with only per-input local information and new matching algorithm (S^3) with computing complexity of O(1), which is more practical in both CIOQ Crossbar and MSM Clos networks.

1 Introduction

With the constantly increasing Internet traffic and the development of broadband access technologies, such as DSL, cable modem and gigabit Ethernet, the next generation routers should support a large number of connection ports for the following two reasons [2][5]. (a) increasing number of Internet accessing points leads to increasing number of input ports and output ports; and (b) Optical transmission technologies such as DWDM is making increasing number of transmitting links available in Internet. The current widely used single stage Crossbar switching fabric, however, can not afford to large number of switching ports for surprising high complexity in switching hardware and scheduling algorithms [3][4][5].

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 209 – 221, 2007.

The Memory Space Memory Clos network, in contrast, is much scalable in switching port number than traditional single Crossbar Fabric, and therefore is causing more and more attention in the next generation of routers. The MSM Clos network itself, however, is not firstly proposed in packet switching domain. In 1953, C. Clos from Bell Systems Labs had proposed the famous Clos network to scale the switching fabric in telephony switches [6]. In circuit switching, more attentions had been paid to blocking property of Clos network to increase call access rates [6][7][8][9]. It is a challenging work to find an efficient and fast scheduling scheme to provide high throughput, starvation-free, acceptable delay, and fairness performance under various traffic conditions for a Clos packet switching network. In [10][11][12], the proposed path-switching scheme and static round-robin (Distro) scheduling algorithm, however, cannot handle various traffic conditions well due to their static nature.

In [1], Shang-Tse Chuang, Ashish Goel etc. studied the speedup problems for CIOQ single Crossbar switch to emulate an OQ switch. They show that a speedup of $2-\frac{1}{N}$ is necessary and a speedup of two is sufficient for this exact emulation. Most interestingly, their result holds for all traffic arrival patterns and is independent with the switching size. The optimal performance of a CIOQ switch urges us to extend the results to MSM Clos network, for the homology in buffering mechanism and the resemblance in architecture between them. We observe and analyze different properties between a single stage Crossbar fabric and a multistage Clos network, and further give the conditions for them to mimic each other. Based on this conditions and non-blocking condition for a reconfigurable Clos network, we provide necessary and sufficient condition for a Clos packet switching network to emulate an OQ switch. Most surprisingly, our result also has the merit of holding for all traffic arrival patterns and being independent with switching size.

However, although the perfect performance of an OQ switch is the target pursued in practical high-speed routers, it has never been achieved in switch with high link speed and large port numbers. This is because the present cell insertion algorithm and matching algorithm have disadvantages of high information complexity and computing complexity in emulating an OQ switch.

We present a method called Per-Input OQ Emulation (PIOE) for MSM CLOS network to emulate an OQ switch based on per-input priority and fairness, which has two merits: (a) without global information exchange among inputs and outputs of the switch, and thus eliminate the information complexity; (b) with an algorithm complexity as low as O(1).

The rest of this paper is organized as follows. In Section II, we introduce some terminology and definitions. In Section III, we describe MSM Clos network Model. In Section IV, we find conditions for the single Crossbar Fabric and the Clos network to mimic each other, and then find the necessary and sufficient condition for Clos network to emulate an OQ switch. In Section V, we put forward a more practical emulation method—the PIOE method, including PVPP-CIP & -CSP and S^3 scheduling algorithm. In Section VI, we will have a conclusion of this paper.

2 Terminology and Definitions

Before proceeding, it will be useful to define some terms used in our presentation. We adopt fixed-length packet concept and call the packets or segment packets 'cells' afterwards. This is common practice in high performance routers [14].

Time slot: Refers to the time taken to transmit or receive a fixed length cell at a link rate of R.

CIOQ Switch: A switch in which there are two stages of buffering on input ports and output ports of an N×N switch. Arriving cells are firstly placed in queues at the input, and then switched to the queues at the output.

OQ Switch: A switch in which arriving cells are placed immediately in queues at the output, where they contend with other cells destined to the same output. The departure order might be FIFO, in which case we call it an FIFO-OQ switch. Other service disciplines, such as WFQ [15], GPS [16], virtual clock [17], and DRR [18] are widely used to provide QoS guarantees. One characteristic of an OQ switch is that the buffer memory must be able to accept (write) N new cells per time slot where N is the number of ports, and read one cell per cell time. Hence, the memory must operate at N+1 times the line rate.

Shadow OQ switch: We will assume that there exists an OQ switch, called the shadow OQ switch, with the same number of input and output ports as the MSM Clos network. The ports on the shadow OQ switch receive identical input traffic patterns and operate at the same line rate as the MSM Clos network.

3 Modeling of MSM Clos Networks

The topology architecture of an MSM Clos network is shown in Fig.1. The basic components in Clos network are switching modules, which can be denoted by X_{nm} with n input ports and m output ports. The three stage Clos network (we only study three stage Clos network in this paper, and it is briefly called Clos network in the rest of this paper) is therefore can be denoted as $[X_{nm}, X_{\mathrm{rr}}, X_{\mathrm{mn}}]$ with r first stage X_{nm} (also called input stage, denoted as IM), m second stage X_{rr} (also called central stage, denoted as CM), r third stage X_{mn} (also called output stage, denoted as OM). The n inputs of each first stage X_{nm} are connected to n input memories of Clos network, and m outputs of X_{nm} connecting to one input of the second stage X_{rr}; r outputs of X_{rr} are connected to one input of r third stage X_{mn}; n outputs of X_{mn} are connected to n output memories of Clos network.

In sense of graph theory, an MSM Clos network can be denoted by a directed graph $C(n,r,m)$, with all switching modules and memories as vertices, and with all connections between all vertices as edges. Then the Clos network can be expressed

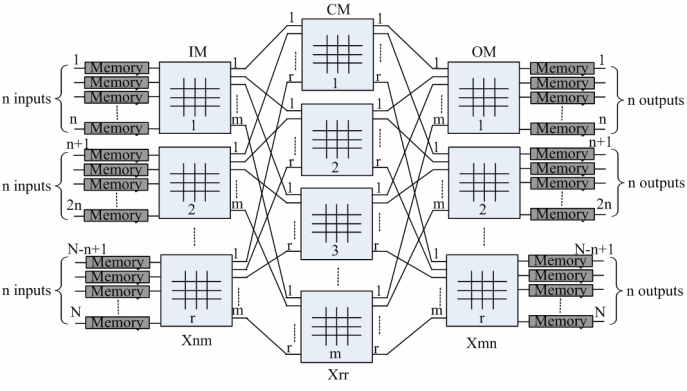


Fig. 1. Topology architecture of an MSM Clos network

as $C(n,r,m)=C(V,E)$, in which V is composed of five incompatible vertex sets and four incompatible edge sets. To be precisely, $V=V_0\cup V_1\cup V_2\cup V_3\cup V_4$

$$\text{where, } V_i=\begin{cases}\{v_1^i,v_2^i,...,v_{nr}^i\} & i=0,4\\ \{v_1^i,v_2^i,...,v_r^i\} & i=1,3\\ \{v_1^i,v_2^i,...,v_m^i\} & i=2\end{cases} \tag{1}$$

$E=E_0\cup E_1\cup E_2\cup E_3$, and E_i is edge sets from V_i to V_{i+1} (i=0, 1, 2, 3),

$$\text{where, } E_i=\begin{cases}\{\vec{xy}\mid x\in\{v_{(j-1)n+1}^0,v_{(j-1)n+2}^0,...,v_{jn}^0\},y=v_j^1,j=1,2,...,\mathrm{r}\} & i=0\\ \{\vec{xy}\mid x\in V_i,y\in V_{i+1}\} & i=1,\ 2\\ \{\vec{xy}\mid y=v_j^3,x\in\{v_{(j-1)n+1}^4,v_{(j-1)n+2}^4,...,v_{jn}^4\},j=1,2,...,\mathrm{r}\} & i=3\end{cases} \tag{2}$$

To describe Clos network conveniently, we shall give some definitions useful in Clos network.

Definition 1. Path — If a directed graph can be tracked from end to end and pass all vertices and edges once, we call the directed graph a Path.

Definition 2. Stable Path — Supposing path p is composed of five vertices and four edges, if all five vertices in p belong to five incompatible vertex sets $\{V_0,V_1,V_2,V_3,V_4\}$, and all four edges in p belong to four incompatible edge sets $\{E_0,E_1,E_2,E_3\}$ of $C(n,r,m)$, respectively, we call path p a stable path in Clos network, and denote it as $\hat{p}$. That is to say, stable path $\hat{p}$ can be defined as follows:

$$\hat{p}\overset{\Delta}{=}\{v_i^0v_j^1v_k^2v_l^3v_q^4,\overrightarrow{v_i^0v_j^1}\in E_0,\overrightarrow{v_j^1v_k^2}\in E_1,\overrightarrow{v_k^2v_l^3}\in E_2,\overrightarrow{v_l^3v_q^4}\in E_3\}$$

Definition 3. Stable Path Set and Stable Path Total Set — Supposing P to be a stable path set with all paths having no compatible edges mutually, if all left stable

paths in $C(n,r,m)$ are always have compatible edges with one of stable paths in P, we call P Stable Path Set in $C(n,r,m)$, and call all possible stable path sets in $C(n,r,m)$ as Stable Path Total Set. We denote stable path set and stable path total set as $S\hat{P}S$ and $S\hat{P}\hat{T}S$, which can be expressed as follows:

$$S\hat{P}S = \{\hat{p}_1, \hat{p}_2, ..., \hat{p}_h, \forall 1 \le i < j \le h, E(\hat{p}_i) \cap E(\hat{p}_j) = \Phi; \forall l > k \ge i \ge 1, E(\hat{p}_i) \cap E(\hat{p}_l) \ne \Phi\}$$

It is very interesting to notice that the stable path set is not unique in $C(n,r,m)$. For example, in stable path set $S\hat{P}S_i$, two stable paths can exchange their vertices in V_2 and exchange their edges in E_1 and E_1, respectively, and then become two new stable paths. Therefore, we get a new path set (i.e. $S\hat{P}S_j$), which is still a stable path set for $S\hat{P}S_j$ shares the same vertices set and edges set with $S\hat{P}S_i$. This property can be explained by architecture of Clos network, in which there are m paths from any input port to any output port. So, there are as many as $(C_m^n)^r \times [(nr)!]$ stable path sets in the stable path total set in $C(n,r,m)$ altogether. Based on the above analysis, we can get two properties of stable path total set by way of graph theory: one is edge exchanging closure, which means a stable path set can become a new stable path set by exchanging any of two stable paths, but this new stable path set is still included by stable path total set. The other is inclusiveness, which means the stable path total set includes all possible stable path sets in $C(n,r,m)$.

Lemmon 1. *The number of stable paths* $SIZE(S\hat{P}S)$ *in a three-stage symmetry Clos network* $C(n,r,m)$ *is* $\min(nr, mr)$.

Proof. From the definition of stable path and property of Clos network, the number of stable paths in Clos network is equal to the minimum value of all four edge sets in $C(n,r,m)$. This can be expressed as follows:

$$SIZE(S\hat{P}S) = \min\{SIZE(E_0), SIZE(E_1), SIZE(E_2), SIZE(E_3)\}$$

Therefore, $SIZE(S\hat{P}S) = \min\{nr, mr, mr, nr\} = \min(nr, mr) = \begin{cases} nr, m \ge n \\ mr, m < n \end{cases}$ ■

Lemmon 2. Supposing the Clos network $C(n,r,m)$ satisfies $m \ge n$, for any integer k ($1 \le k \le nr$) and any vertex pair set $M_f = \{(a_i, b_i) \mid i = 1,2,...,k\}$ from vertex set $A = \{a_1, a_2, ..., a_k\} \subset V_0$ to $B = \{b_1, b_2, ..., b_k\} \subset V_4$, there always stable path set $S\hat{P}S_i \subset S\hat{P}\hat{T}S$ connecting k vertex pairs in M_f.

Proof. Because integer k satisfies $k \le nr \le mr$, from Lemmon 1 we know that $k \le SIZE(S\hat{P}S)$ and therefore $k \le SIZE(E_0) = SIZE(E_3)$, and $k \le SIZE(E_1) = SIZE(E_2)$ respectively. So we always can find four edge sets composed of k edges

from E_0, E_1, E_2 and E_3 to buildup k stable paths to connect k vertex pairs in M_f. From property of inclusiveness of stable path total set of $C(n,r,m)$, there always stable path set $S\hat{P}S_i \subset S\hat{P}\hat{T}S$ connecting k vertex pairs in M_f. ■

We notice here that Lemmon 2 is equivalent to the reconfigurable non-blocking condition of the Clos network, which had been proved in prior results [6][9].

4 Emulating an OQ Switch by MSM Clos Network

The non-blocking condition, in Lemmon 2 in Section III, guarantees any number of vertex pairs connected by a stable path set in the Clos network. Intuitively, this is quite similar to the non-blocking property of a single stage Crossbar Fabric.

Lemmon 3. Supposing MSM Clos network $C(n,r,m)$ satisfies $m \geq n$, for single stage CIOQ Crossbar Fabric $C(nr \times nr)$, we say MSM Clos network $C(n,r,m)$ mimics $C(nr \times nr)$.

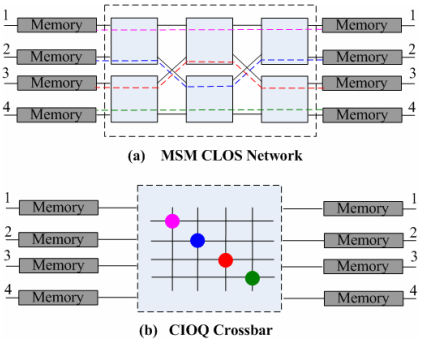


Fig. 2. A 4×4 CIOQ Crossbar mimics the MSM Clos network

We do not mean to give strict proof of Lemmon 3 here, but informally provide an analysis by way of an example in Fig.2. There are four input ports and four output ports for both MSM Clos network and CIOQ Crossbar. For the same port pair set $\{(1,1),(2,2),(3,3),(4,4)\}$, we can either find a stable path set $S\hat{P}S_4$ in MSM Clos network, or find matching matrix $\tilde{M}_4$ in CIOQ Crossbar, to find stable paths or configure crosspoints to set up connections between all four port pairs. Intuitively, stable path set $S\hat{P}S_4$ and matching matrix $\tilde{M}_4$ have played the same role in the switching fabrics. In this sense, we say a non-blocking (i.e. $m \geq n$) MSM Clos network and CIOQ Crossbar mimic each other, and call the stable path set is equivalent to the matching matrix $S\hat{P}S \Leftrightarrow \tilde{M}$.

$$S\hat{P}S_4 = \{\hat{p}_1, \hat{p}_2, \hat{p}_3, \hat{p}_4\} = \begin{Bmatrix} \hat{p}_1 = v_1^0 v_1^1 v_1^2 v_1^3 v_1^4 \\ \hat{p}_2 = v_2^0 v_1^1 v_2^2 v_1^3 v_2^4 \\ \hat{p}_3 = v_3^0 v_2^1 v_1^2 v_2^3 v_3^4 \\ \hat{p}_4 = v_4^0 v_2^1 v_2^2 v_2^3 v_4^4 \end{Bmatrix} \quad \tilde{M}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Theorem 1. *The necessary and sufficient condition for an MSM Clos network $C(n,r,m)$ to emulate an OQ switch is with speedup of S, where S satisfies:*

$$S \geq \begin{cases} \frac{n}{m} \times (2 - \frac{1}{mr}) & m < n \\ 2 - \frac{1}{nr} & m \geq n \end{cases} \tag{3}$$

Proof: (a) For the case of $m \geq n$, we can refer to Lemmon 3 and regard the MSM Clos network $C(n,r,m)$ mimics a CIOQ Crossbar fabric with port number $C(nr \times nr)$. So, the results in [1] still hold where $N = nr$, and then we get $S = 2 - \frac{1}{nr}$ for an MSM Clos network to emulate an OQ switch.

(b) For the case of $m < n$, from Lemmon 1 we know that the number of stable paths $SIZE(S\hat{P}S)$ in a stable path set $S\hat{P}S$ is $\min(nr, mr)$. Therefore, there are at most mr stable paths in the MSM Clos network. Fig.3 shows an example of an MSM Clos network with $n = 4$, $m = 2$, $r = 2$ and a CIOQ Crossbar with port number 8×8 and switch fabric of 4×4. Because there is at most 2×2=4 stable paths in the MSM Clos network, regarding the former case of $m \geq n$, four out of eight input ports can emulate an OQ switch with a speed up of $(2 - \frac{1}{4})$. Thus, if we divide the input ports into two groups (i.e. the light colored input ports group and the dark colored input ports group in Fig.3), the emulation can be divided into two phases (i.e. the light colored Phase 1 and dark colored Phase 2) for each group. Therefore, the four input ports in Group 1 can emulate an OQ switch in Phase 1 with speedup of $(2 - \frac{1}{4})$, while the left four input ports in Group 2 can also emulate an OQ switch in Phase 2 with speedup of $(2 - \frac{1}{4})$. Then the total speedup for the MSM Clos network emulating an OQ switch is $(2 - \frac{1}{4}) + (2 - \frac{1}{4}) = 2 \times (2 - \frac{1}{4})$. The proof of any case of $m < n$ is a straight forward extension of the example with $n = 4$, $m = 2$, $r = 2$, where the speedup needed by an MSM Clos network is proportional to $\frac{n}{m}$ and $(2 - \frac{1}{mr})$. According to pigeon hole principle and Constraint Set Principle, with a speedup of $\frac{n}{m} \times (2 - \frac{1}{mr})$, an MSM Clos network can emulate an OQ switch. ■

What deserves more attention here is when the speedup S is not an integer in MSM Clos network. We can use the smallest integer $\overline{S} = \lceil S \rceil$ bigger than S as speedup in some time slots, while using $(\overline{S} - 1)$ as speedup in the other time slots in a circle. To be more precise, supposing $S = \overline{S} - \frac{F}{G} (F < G)$, let a circle length to be G time slots. We adopt a

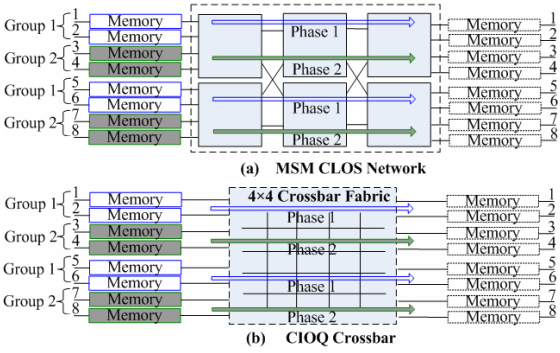


Fig.3. Emulating an OQ switch when $n = 4, m = 2, r = 2$

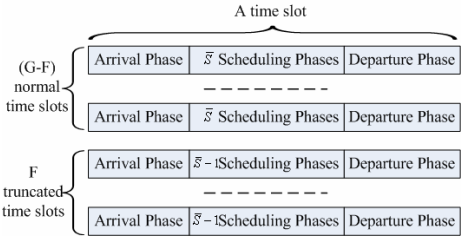


Fig.4. Normal time slots and truncated time slots in a circle

speedup of $\overline{S}$ in $G-F$ time slots and a speedup of $(\overline{S}-1)$ in F time slots. Meanwhile, we call these F time slots as truncated time slots. Fig.4 shows the normal time slots and truncated time slots in a circle.

5 Per-Input OQ Emulation (PIOE) by MSM Clos Network

The current cell insertion algorithms and matching algorithms have the following disadvantages in emulating an OQ switch:

(a) High Information complexity: In CIOQ emulation with an OQ switch, Shang-Tse Chuang had relied on the stable matching during the two scheduling phases and CCF (Critical Cell First) cell insertion policy during arriving phase in each time slot [1]. The stable matching needs to know the output priority list which relies on information of output order of all cells on the input side. The CCF algorithm needs to know the number of cells with a higher priority on the output side. Meanwhile, both the stable matching algorithm and the CCF cell insertion algorithm need to know the departure time of each cell, which is calculated based on the priority information of all cells in both input side and output side. Thus an input has to communicate with all the other inputs and outputs to obtain the information needed by each cell during each time slot, but this is too difficult to implement not only for real time information exchange, but also for modularization design, which never means to have so many extra communications beyond normal packet forwarding.

(b)High Computing complexity: The stable matching that CIOQ switch needs to find in each scheduling phase can take as many as $O(N^2)$ iterations, which is proved to be equivalent to the number of cells buffering on the input side[Gale and Shapely [13]. Unfortunately, as the increase of link speed as well as the switch port numbers, schedulers will have to make more decisions (i.e. there will be more iterations to guarantee a maximal matching and hence obtain high performance) in a more urgent time slot (i.e. the time will be only a fraction of the time slot in low link speed case). In this case, the matching algorithms available in switch with low link speed and less switch ports just can not be applied to new generation routers.

To overcome these disadvantages, we discuss a method for MSM CLOS switch to emulate an OQ switch with an acceptable or predictable performance degrading (i.e. emulation only based fairness of per input port). For example, the method (a) does not need information exchange among inputs and outputs of the switch, and thus eliminate information complexity; (b) has an algorithm complexity as low as O(1).

We observe that in an OQ switch cells are inserted into and scheduled from input queues based on the global priority principles (i.e. WFQ [15], Strict Priority [18], or FIFO among all cells from all input ports), and this leads to cell insertion algorithm and cell scheduling algorithm requiring global information (cell queueing states among all inputs and outputs). The emphasis on the priority order in an OQ switch, however, is almost meaningless for cells coming from different links and buffered in different input queues, because they have almost no relations with each other. Based on this fact, we present a method for MSM CLOS network to emulate an OQ switch based on per-input priority and fairness, which only uses information locally available on each input. We call this method Per-Input OQ Emulation (PIOE).

In the method of PIOE, we organize cell queueing in a way like Virtual Output Queueing (VOQ), but use a wide class of queueing policies such as WFQ and Strict Priority queueing in each VOQ queue of each separate input, just like an OQ switch does. The PIOE method comprises Per-VOQ Per-Priority based Cell Insertion Policy (PVPP-CIP), Cell Scheduling Policy (PVPP-CSP), and S^3 scheduling algorithm.

To describe more precisely, we shall give two definitions as follows:

Definition 4. VOQ Queue — *In each input, cells are buffered in different queues according to their output port number, and we denote them as* Q_{ij} *, where* $i, j \in \{1,2,...,nr\}$ *for am MSM Clos networks* $C(n,r,m)$*. It is easy to know that there are* nr *VOQ queues in an input port and* $(nr)^2$ *VOQ queues in all input ports.*

Definition 5. VOQ Priority Queue—In each VOQ queue, cells are buffered in different queues according to their priority number, and we denote them as $Q_{p_k}(k = 1,2,...,K)$, where K is the number of priorities supported by routers.

5.1 PVPP Cell Insertion Policy (CIP) of PIOE

PVPP Cell Insertion Policy: Supposing that cell X_{ijk} arrives at input port i and is destined for output port j and has a priority number k. Upon arrival X_{ijk} is inserted to

the end of VOQ priority queue Q_{p_k} of the VOQ queue Q_{ij}. Because cells are inserted into each input port based on Per VOQ and Per Priority, we call this cell insertion policy PVPP-CIP.

5.2 PVPP Cell Scheduling Policy (CSP) of PIOE

PVPP Cell Scheduling Policy: Supposing that X_{ijk} represents cells buffered in VOQ priority queue Q_{p_k} of the VOQ queue Q_{ij}. Upon departure cell X_{ijk} is exported in sequence of VOQ queues and VOQ priority queues. Because cells are scheduled out of each input port based on Per VOQ and Per Priority, we call this cell scheduling policy PVPP-CSP.

PVPP-CIP & -CSP of PIOE are shown in Fig.5. In the left dashed frame, PVPP-CIP is composed of two phases: one is the VOQ dispatcher, where cells are classified into each VOQ queue according to their output port number; and the other is the Priority dispatchers, where cells are classified into each VOQ priority queue according to their priority number. In the right dashed frame, PVPP-CSP is also composed of two phases: one is the priority schedulers to export cells from different VOQ priority queues; the other is the VOQ scheduler, where cells from different VOQ queues are exported to the output ports.

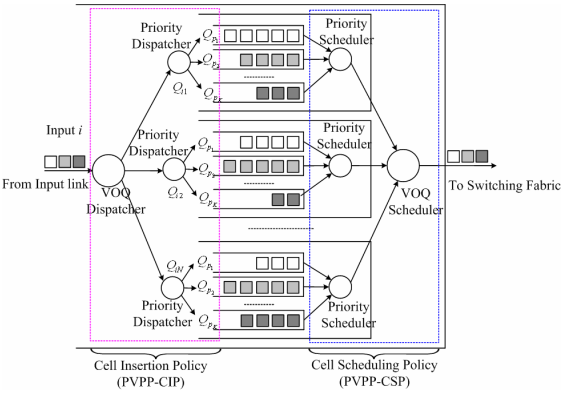


Fig. 5. PVPP Cell Insertion Policy and Cell Scheduling Policy of PIOE

5.3 Queueing Principle Analysis of PVPP-CIP and –CSP

(a) FIFO queueing principle emulated by PVPP-CIP and –CSP: In FIFO queueing principle, it does not need classify cells based on priorities; therefore the priority dispatche and the priority scheduler only maintain one priority queue (i.e. the highest priority queue Q_{p_K}). Thus each priority dispatcher just writes cells to the end of Q_{p_K}, while each priority scheduler just reads cells from head of Q_{p_K}.

(b) Strict Priority and WFQ queueing principle emulated by PVPP-CIP and –CSP: These two kinds of queueing principles are sensitive to cell's priority. In PVPP-CSP,

cells are written to priority queues by each priority dispatcher, which is similar for both Strict Priority and WFQ. In PVPP-CSP, however, priority schedulers read cells in different ways. Cells in highest priority queues that are not empty are prior to cells in lower priority queues in Strict Priority queueing principle. While in WFQ queueing principle, each priority queue is endowed with a weight value $w_k (k = 1,2,...,K)$, and the scheduling times in each scheduling circle are proportional to each w_k.

5.4 S^3 Matching Algorithmin PIOE

To overcome the high computing complexity of stable matching in [13] (i.e. their solution has a complexity of $O(N^2)$), and overcome the high information complexity of *GBVOQ* in [1] (i.e. it needs global state information), we design a simple and practical matching algorithm based on per input port fairness.

Definition 6. Vertex Matching—If a pair of vertices belongs to V_0 and V_4 of Clos network, respectively, we call the vertex pair a vertex matching, and denote it as $M(v_i^0, v_j^4)$, where $i, j \in (1,2,...,nr)$.

There are $(nr)^2$ pairs of vertex matching altogether in $C(n,r,m)$; we can further divide them into nr incompatible groups, each of which includes all input vertices and all output vertices of $C(n,r,m)$. We call each group a Stable Vertex Matching (*SVM*), and further call all nr groups the *Comple SVM Set* (*CSS*) for including all possible $(nr)^2$ pairs of vertex matching. The definitions of *SVM* and *CSS* are as follows:

$$SVM_i = \{M(v_1^0, v_{(i) \bmod (nr)}^4), M(v_2^0, v_{(1+i) \bmod (nr)}^4), ..., M(v_{nr}^0, v_{(nr-1+i) \bmod (nr)}^4)\}, i = (1,2,...,nr)$$

From Lemmon 2, we can always find a stable path set $S\hat{P}S$ for each *SVM* in *SPTS* of $C(n,r,m)$, and therefore can find nr SPS_i $(i = 1,2,...,nr)$ for all *SVM* in *CSS*.

Definition 7. Mapping Table from *SVM* to *SPS*—In non-blocking Clos network, we call each pair of each *SVM* to each *SPS* a mapping table and denote it as $M_{(SVM,SPS)}$.

$$M_{(SVM,SPS)} = \{(SVM_i, SPS_i), i = 1,2,...,nr\}$$

Based on the mapping table our matching algorithm of PIOE is as follows:

Stable SVM SPS (S^3) **matching algorithm:** During i^{th} of matching of PIOE, we choose $(SVM_{i \bmod nr}, SPS_{i \bmod nr})$ in $M_{(SVM,SPS)}$ as matching of inputs and outputs.

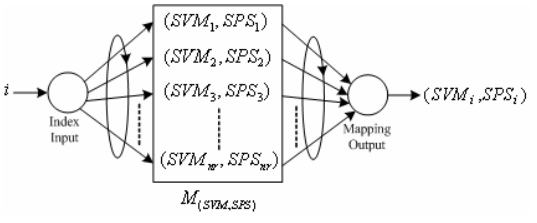


Fig. 6. S^3 matching algorithm of PIOE

Obviously, we can see there are two properties for S^3 matching algorithm: one is low computing complexity of $O(1)$, for just requiring one lookup in mapping table; the other is absolute fairness among all inputs and outputs, for in a round of nr matching, all pairs of input and output appear once and only once. That is to say, all inputs are absolutely fair in S^3 matching algorithm.

6 Conclusions

We have studied MSM Clos packet switching network in this paper. Firstly, we discussed the modeling method of MSM Clos network based on graph theory, and put forward stable path set and non-blocking property of Clos network (Lemmon 1 and Lemmon 2). Based on the similarity of single stage CIOQ Crossbar and multistage MSM Clos network, we discussed the condition for them to mimic each other (Lemmon 3). Then we extent the results in CIOQ Crossbar emulation an OQ switch to MSM Clos networks (Theorem 1). To overcome the disadvantages of cell insertion algorithm and matching algorithm in CIOQ Crossbar emulating an OQ switch, we provided a PIOE method with PVPP-CIP & -CSP with low information complexity and S^3 matching algorithm with computing complexity of $O(1)$.

Acknowledgements

This work was supported in part by the grants from National Basic Research Program of China (973 Program) with No.2007CB307102 and National Hi-tech Research and Development Program of China (863 program) with No.2005AA121210. We thank several members of Information Engineering Institute for their technical suggestions, including Peng Yi, Yufeng Li and Yang Li, and the anonymous reviewers for their constructive comments and suggestions.

References

1. Chuang, S.T., Awadallah, A., McKeown, N., Prabhakar, B.: Matching output queueing with a combined input and output queued switch. IEEE Journal on Selected Areas in Communicaions 17, 1030–1039 (1999)
2. Chao, H.J.: Next generation routers. IEEE Proceeding 90(9), 1518–1558 (2002)
3. McKeown, N.: The iSLIP Scheduling Algorithm for Input-Queued Switches. IEEE/ACM Trans. on Networking 7(2) (1999)
4. McKeown, N., Anntharam, V., Walrand, J.: Achieving 100% Throughput in an input-queued switch. In: Infocom 1996 (1996)
5. Wang, F., Hamdi, M.: Analysis on the Central-stage Buffered Clos-network for packet switching. In: IEEE International Conference on Communications (2005)
6. Clos, C.: A Study of Non-Blocking Switching Networks. Bell Systems Technical Journal, 406–424 (1953)
7. Tsai, K.H., wang, D.W.: Lower Bounds for Wide-sense Non-Blocking Clos Network. In: Taipei 1998. Computing and Combinatorics, Springer, Berlin, pp. 213–218 (1998)

8. Lee, T.T., To, P.P.: Non-Blocking Routing Properties of Clos Networks. In: Advances in switching networks, Amer. Math. Soc., Providence, RI, pp. 181–195 (1998)
9. Lin, G.H., Du, D.Z., Wu, W., Yoo, K.: On 3-Rate Rearrangeability of Clos Networks. In: Advances in switching networks, Amer. Math. Soc., Providence, RI, Princeton, NJ, pp. 315–333 (1998)
10. Lee, T.T., Lam, C.H.: Path Switching - A Quasi-Static Routing Scheme for Large-Scale ATM Packet Switches. IEEE J. Select. Areas Communications. 15, 914–924 (2002)
11. Pun, K., Hamdi, M.: Distro: A Distributed Static Round-Robin Scheduling Algorithm for Bufferless Clos-Network Switches. IEEE GLOBECOM (2002)
12. Chao, H.J., Deng, K-L., Jing, Z.: A Petabit Photonic Packet Switch (P3S). IEEE INFOCOM 2003 (2003)
13. Gale, D., Shapley, L.S.: College Admissions and the Stability of Marriage. American Mathematical Monthly 69, 9–15 (1962)
14. Iyer, S., Awadallah, A., McKeown, N.: Analysis of a Packet Switch with Memories Running Slower than the Line Rate. In: IEEE Infocom 2000 (2000)
15. Demers, A., Keshav, S., Shenker, S.: Analysis and Simulation of a Fair Queueing Algorithm. J. Internetworking: Research and Experience, 3–26 (1990)
16. Parekh, A., Gallager, R.: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case. IEEE/ACM Trans. Networking 1, 344–357 (1993)
17. Zhang, L.: Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. ACM Trans. Comput. Syst. 9(2), 101–124 (1990)
18. Shreedhar, M., Varghese, G.: Efficient Fair Queueing Using Deficit Round Robin. In: Proc. ACM SIGCOMM, pp. 231–242. ACM Press, New York (1995)

Measurement of High-Speed IP Traffic Behavior Based on Routers

Xiangjie Ma, Junpeng Mao, Yuxiang Hu, Julong Lan, Lian Guan, and Baisheng Zhang

Information Engineering Institute, PLA Information Engineering University
National Digital Switching System Engineering & Technological Research Center
Zhenzhou, Henan, 450002, P. R. China
maxiangjie100@163.com,
{mxj,mjp,hyx,ljl,gla,zbs}@mail.ndsc.com.cn

Abstract. IP traffic behavior is becoming increasingly complicated and complex with the appearance of new applications and new protocols in Internet. We give a design to implement fine-granularity and configurable distributed measurement method of multi-protocol traffic behavior. Our study shows that probing points can be distributed to different the Functional Processing Module (FPM) along the traffic path through the router, and each FPM can implement configurable and protocol-sensitive collection of packet information with accurate timing stamps. We develop a novel traffic mirroring method to avoid affecting normal packet processing in case of some occasional failures. Our experiments based on practical implementation of measurement in a router show that less than 15% of total logic resources and less than 20 nanoseconds of timing precision can be achieved by our methods, which is more efficient and accurate than Special Packet capturing Card with an acceptable cost.

1 Introduction

In order to run networks efficiently, network administrators need to understand how their networks are used or misused. Therefore, traffic measurement is an important method to monitor the traffic mix, especially on important links.

Traffic measurement, however, is a challenging area of research. Most routers report traffic measurement data in NetFlow format [9] that aggregates all packets belonging to the same flow into a flow record, which is becoming increasingly fragile for not adapt to most traffic mixes. In traditional low-speed measurement techniques, Special Packet capturing Cards (SPC) are put on the interested links and observe packets passing through the link. We can analyze the data samples to compute performance metrics [1-5], study protocol behavior [3, 6], and understand the dynamics of traffic demands in the network [7, 8]. To implement measurement of high speed IP traffic based on routers, we have designed a FPM (Functional Processing Module) measurement method, which can be further divided into two classes: i-FPM and d-FPM according to the directions of IP traffic through a router. Then we provide a distributed and robust measurement system of IP traffic behavior, which is

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 222 – 232, 2007.

configurable and protocol-sensitive with high accurate timing stamps. By practical implementation and experiments based on routers, we find that this measurement method is an efficient hardware implementation method compared with its software counterpart, which occupies less than 15% of total logic resources and has cell timing stamps as accurate as 20 nanoseconds.

The rest of this paper is organized as follow. In Section II, we introduce some terminology and definitions. In Section III, we provide measurement method of IP traffic behavior based on routers. In Section IV, we At the end of this paper is the conclusion of our work.

2 Terminology and Definitions

Before proceeding it will be useful to define some terms used throughout this paper:

Router: Refers to a standard packet' buffering and forwarding router in Internet, which has N×N ports connecting to outside links.

Cell: Refers to a fixed-length packet, though not necessarily equal in length to a 53-byte ATM cell. Although packets arriving to switch may have variable length, for the purposes of this paper we will assume that they are segmented and processed internally as fixed length cells. This is common practice in high-performance switches; variable length packets are segmented into cells as they arrive, carried across the router as cells, and reassembled back into packets before they depart.

Input port: Refers to the incoming end of packets for a router, where packets will be classified, buffered and updated according to the result of looking up the forwarding table. The traffic captured by the input port can reflect the original characteristics of the incoming links of the router.

Output port: Refers to the departing end of packets for a router, where packets will be buffered, scheduled by priority, and pooled into one link. The traffic captured by the output port can reflect the influence caused by the processing of the router.

Time slot: Refers to the time taken to transmit or receive a fixed length cell at a link rate of R, where R represents the speed of each input port and output port.

VOQ queues: Refers to the N sub-queues of each input port, where cells will be buffered according to their destination output port to avoid the problem of HOL (Head of Line). To describe the arriving traffic of each input port more accurately, this paper supposes that the arriving traffic of each input port will firstly separated into N different sub-traffic, which will be buffed into N VOQ queues, respectively.

3 Measurement of IP Traffic Behavior

We are interested in deploying measurement of IP traffic behavior to incorporate flexibility of hardware processing and reduce complexity of software processing in a router. Thus, our measurement study focuses on the following aspects: (1) more parallelism to permit higher linking speed and more protocol characteristics; (2) more

measuring functions implemented by hardware to relax the overheads of software; (3) more accurate timing stamps of cells on the basis of hardware clocks to increase precision than measurement by software. In this section, we describe the architecture of universal routers and present our findings.

3.1 Architecture of Universal Router

The basic architecture of universal routers is shown in Figure 1. The building blocks for a universal router include line cards, forwarding units, scheduling units, input switch units, System Optical Backboard (SOB), System Internal Communication (SIC) and System Central Processor (SCP).

Firstly, packets enter the router from each line card, where they will be segmented into equal length cells and classified into data cells and protocol cells. Secondly, the protocol cells destined to this local router will be sent to SCP for further processing through SIC, while the data cells will be transmitted to forwarding units, where they are stamped with a destination output number according to the result of looking up the forwarding tables. Thirdly, the stamped cells will be balanced to different input switch units and switched to their destination output scheduling unit. Lastly, the switched cells will be buffed into different queues according to their priorities and scheduled with the order of corresponding scheduling policy and delivered from the output line card.

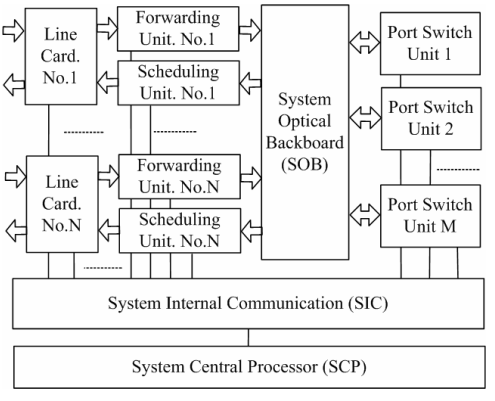


Fig. 1. System architecture of a universal router

3.2 Distributed Measurement Methodology by FPM

We conduct a passive online measurement study of IP traffic behavior based on routers. For the purposes of our measurement, we analyze the detailed implementation methods of each Functional Processing Module (FPM). We choose conduct our measurement on distributed FPM because (i) all the FPMs on the same side of a router have the same characteristics of IP traffic behavior for all the processing is completed at the speed of line rate; and (ii) it is easier to accommodate only a fraction of measurement by each FPM for occupying less hardware logical resources than the centralized implementation methods.

As shown in Figure 1, IP traffic are processed and forwarded among different FPMs in two directions: the incoming direction with FPMs of line cards and forwarding units, and departing direction with FPMs of scheduling units and line cards. The FPMs in the same directions have identical characteristics for sharing the same processing speed of line rates. So, we can capture the incoming IP traffic for inputting links on the incoming FPMs (i-FPM) and capture the departing IP traffic for outputting links on the departing FPMs (d-FPM). Obviously, forwarding units and scheduling units belong to i-FPM and d-FPM, respectively. Line cards incorporate incoming and departing traffic simultaneously, so it is not only belongs to i-FPM, but also belongs to d-FPM. Input switch units accommodate all the incoming and departing traffic and have the heaviest switching loads, so we do not dispatch any traffic measurement tasks to them.

Line cards provide LAN, WAN or POS serial data-linking layer processing in line rate, which complete the classifications of cells of different protocols, including IPv4, IPv6 and MPLS, etc. In addition, line cards also have the functions of supporting ARP and neighbor-discovering protocols.

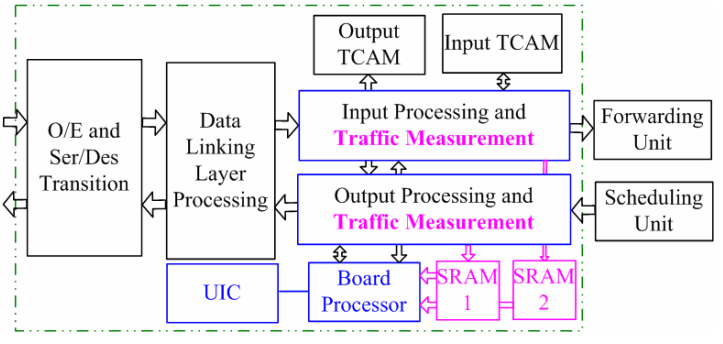


Fig. 2. Line Card Structure with traffic measurement

The internal functional modules of line cards are shown in Figure 2. The O/E and Ser/Des Transition module and Data Linking Layer Processing module complete data interface processing and data linking layer processing, respectively. In the data incoming direction, the Input Processing module segments packets into cells and classifies them according to the address list stored in the Input TCAM module. In the data departing direction, cells are combined and framed into Ethernet format packets according to the MAC address stored in the Output TCAM. Meanwhile, the traffic measurement modules are denoted in pink color, while a mixing function of normal cell processing and measurement are denoted in blue color.

The Forwarding Units structure with traffic measurement are shown in Figure 3, which mainly complete IP layer processing including IP unicasting, multicasting, and MPLS processing. The TCAM and SRAM (black-colored) module stores the forwarding table for all the relative protocols. Meanwhile, the traffic measurement modules are denoted in pink color, while a mixing function of normal cell processing and measurement are denoted in blue color.

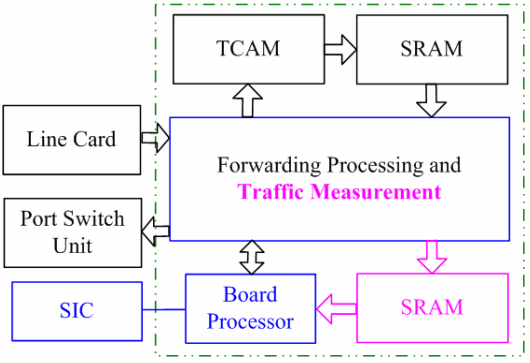


Fig. 3. Forwarding Units structure with traffic measurement

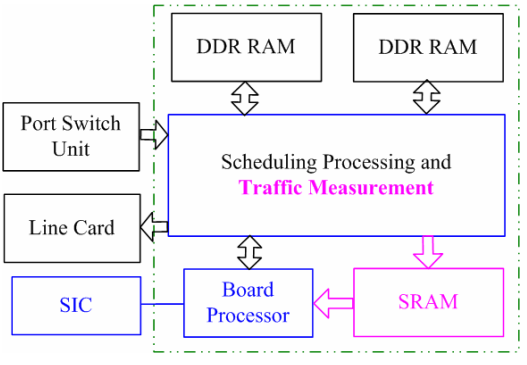


Fig. 4. Scheduling Units structure with traffic measurement

Figure 4 gives the structure of Scheduling Units with traffic measurement functions, and the units aim to provide multiple IP priority queues and allocate different bandwidth to different priority queues. The traffic measurement modules are denoted in pink color, while a mixing function of normal cell processing and measurement are denoted in blue color.

3.3 A Novel Protocol-Sensitive Measurement Method

To achieve the flexibility of hardware measurement without inducing potential instability and occupying too much logical resources, we explore a novel protocol-sensitive measurement method, which has the following features: (1) it can be implemented in distributed manner to alleviate the overheads put on each FPM and increase robustness of data measurement; (2) it can be configurable to measure different protocol data and complete corresponding computations of our foregoing IP traffic model; (3) it should not insert the pipeline of normal data processing directly, which may be affected in case of occasional failure; (4) it can append cells with high-accurate timing stamps and send them periodically to the System Central Processor (SCP) for additional analytical purpose.

As discussed in the preceding sub-section, the measurement method adopts i-FPMs including the input line cards and the forwarding units to measure protocol traffic of the incoming direction, and adopts d-FPMs including the output line cards and the scheduling units to measure protocol traffic of the departing direction. Our designation and assignation method is shown in Table 1. Line cards are assigned to measure various routing protocols and ARP traffic because they have advantages to classify and report them to the SCP. The forwarding units and the scheduling units are used to measure IPv4, IPv6, and MPLS protocol data as they have corresponding special processing modules.

To satisfy feature (3) described at the beginning of this sub-section, we adopt a new traffic mirror method shown in Figure 5. This improvement can isolate the measurement function from normal data processing and reduce influences to the lowest level.

Table 1. Protocol traffic measured by each i-FPM and d-FPM

i-FPMs		d-FPMs	
Line Card (input part)	For-warding Units	Sche duling Units	Line Card (output part)
RIP, RIPng, OSPF, IS-IS, BGP4, PIM-SIM, ARP	IPv4, IPv6, MPLS	IPv4, IPv6, MPLS	RIP, RIPng, OSPF, IS-IS, BGP4, PIM-SIM, ARP

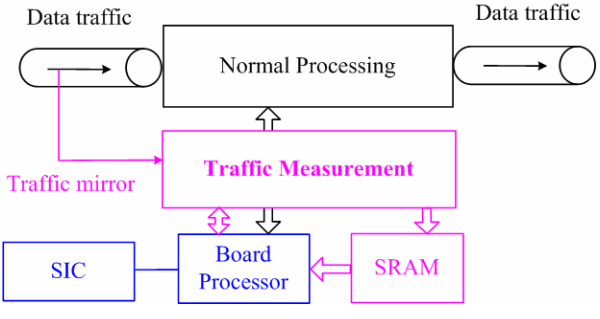


Fig. 5. Traffic mirror method in each FPM

According to practical applications in traffic measurement, only the traffic of one protocol is analyzed in a period of time for a single port. So we explore a measurement method that will not only calculate the traffic behavior of all protocols but also report the traffic arriving data with accurate timing-stamp according to the protocol configured by SCP. The PPP protocol packet format used by SCP to configure traffic measurement of each FPM is shown in Figure 6.

1byte	1byte	1byte	2byte	≤1500byte	1byte
flag 7E	Address FF	Control 03	Protocol	Data	flag 7E

Fig. 6. PPP protocol packet format used to configure measurement

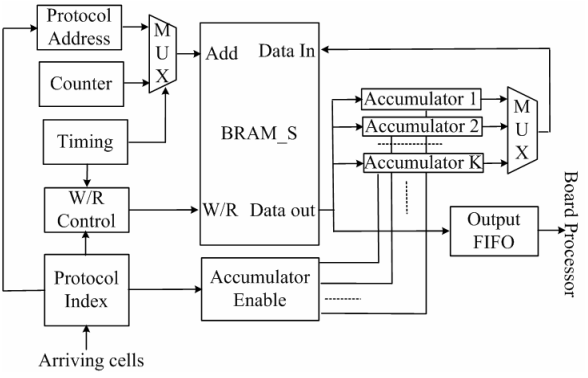


Fig. 7. Protocol traffic behavior measurement

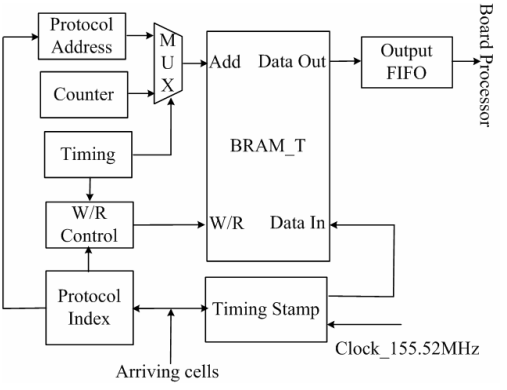


Fig. 8. High accurate timing stamp measurement

Figure 7 illustrates the basic computing methods for various protocol cells. There are mainly two operation processes in this method: one is the online protocol computing process, which includes modules such as Protocol Index, Protocol Address, K Accumulators, and Accumulator Enable; the other is periodically reporting process, which includes modules such as Timing, Counter, and Output FIFO. According to the model of IP traffic behavior described in section III, the basic for computing and judging the average arriving rate, the admissible traffic, the uniform traffic, the balanced traffic and the slotted traffic is the arriving traffic Aij(t) (i=1, 2, …, N; j=1, 2, …, N) of each protocol. So, the BRAM_S module is segmented into N blocks corresponding to N output ports and each block is segmented into K sub-blocks corresponding to K protocols that are to be analyzed. Therefore, each kind of protocol of

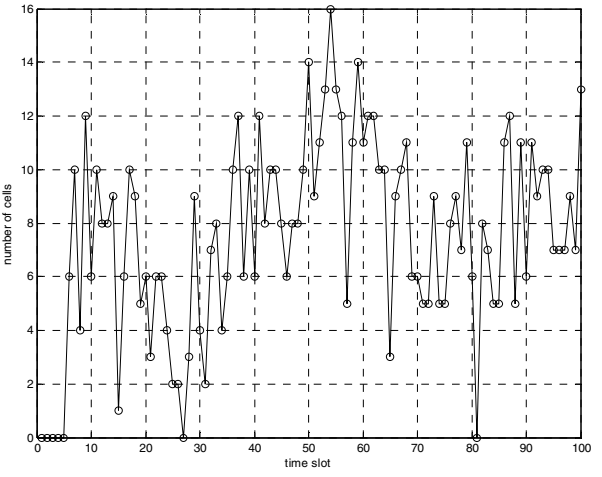


Fig. 9. Arriving traffic of the router over 100 time slots

each port is configured with a separate buffering area in the BRAM_S and its address is coded by the Protocol Index module. Firstly, the Protocol Index module is triggered on each arrival of cells and switches the protocol type to its address in BRAM_S, and simultaneously configures the W/R Control module to the state of "read" and enables the corresponding Accumulator through the Accumulator Enable module. Secondly, the data Aij(t) corresponding to the protocol address is sent to the enabled Accumulator. Thirdly, the accumulating result {Aij(t) +1} is written to BRAM_S. The reason why we have K Accumulators is to support pipeline operations, that is to say this method can support continuous arriving cells accumulations.

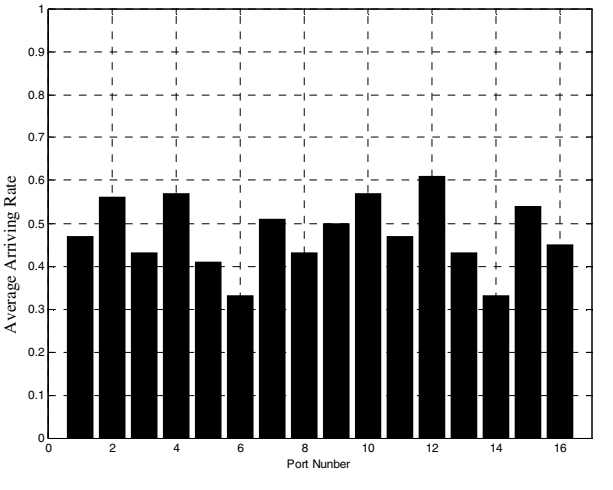


Fig. 10. Average arriving rates of 16 inputs over 100 time slots

The periodically reporting process is used to send the calculating results to SCP at a configured timing interval T stored and counted in the Timing module. Each reporting results are firstly stored in the Output FIFO and then sent to SCP through Board Processor. After each reporting, all blocks of BRAM_S are cleared to zero.

To satisfy feature (4) we deplore a high accurate timing stamp measurement method shown in Figure 8. This measurement method also has two processes: one is the timing stamps appending process and the other is the periodically reporting process. The periodically reporting process is identical to that of the protocol traffic behavior measurement. The timing stamps appending process includes a Timing Stamp module, which is clocked by a clock of 155MHz frequency. Because each delay of appending timing stamps to a cell is no more than 3 clock period, the accuracy for this measurement method is less than 20 nanoseconds. What's more, we constrain the occupation of logical resources to only 15% in our practical implementations.

4 Measurement Results

To verify the measurement method in Section III, we have analyzed and modeled of the real-time data captured by our designed router with 16×16 port numbers built by i-FPMs and d-FPMs. Figure 9 describes total arriving traffic for all 16 input ports of the router over 100 time slots, and we can see from the graph that there are 4 to 12 cells arriving in most time slots, which means there are 4 to 12 input ports with cells arriving in these time slots. The average arriving rates for all 16 input ports are shown in Figure 10, and we can see that there are 10 input ports with an average arriving rate less than 0.5 and all less than 0.7. To illustrate the unbalancing and burst feature the arriving traffic behavior, we also give Figure 11 and Figure 12. From Figure 11 we can see that the Unbalanced Sizing U in 96% time slots is less than 0.1, and only time

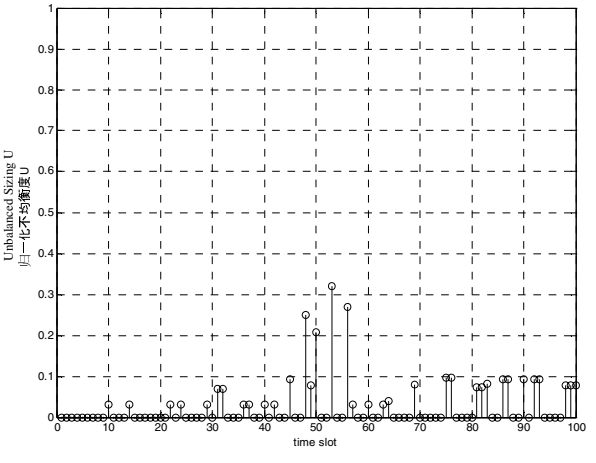


Fig.11. Unbalanced Sizing U of 16 inputs over 100 time slots

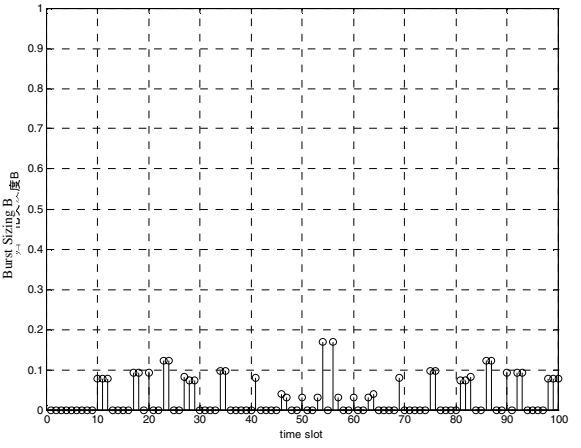


Fig.12. Burst Sizing B of the 1st inputs over 100 time slots

slot of number 48, 50, 53, and 56 have a value of U more than 0.1 but still less than 0.35. By Figure 12, we can clearly see that the Burst Sizing B of the first input port in 94% time slots is less than 0.1, while only the values of B in time slot of 23, 24, 54, 56, 86, 87 are higher than 0.1 but less than 0.2. Clearly, over these 100 time slots, the router has benign arriving traffic in the balancing and burst sense.

5 Conclusions

In this paper, we conducted a study of measurement of IP traffic behavior based on routers. According to the actual distribution and traveling path of IP traffic in routers, we discussed the measurement method of IP traffic behavior. Compared with other measurement methods, our method has the following merits: (1) it adopts distributed implementation mechanism with i-FPM in the incoming traffic direction and d-FPM in the departing traffic direction; (2) it adopts a traffic mirror mechanism to avoid affecting normal data processing; (3) it is configurable to measure protocol-sensitive traffic and can complete corresponding computations of our foregoing IP traffic model; (4) it can append requested protocol cells with high accurate timing stamps and report them to System Central Processor (SCP) for further analytical needs.
In our practical designations and experiments, we found that our traffic behavior model can be easily implemented in hardware and that our traffic measurement method occupies less than 15% of total logical resources of each FPM and have a timing stamp as accurate as 20 nanoseconds.

Acknowledgement

This work was supported in part by the grants from National Basic Research Program of China (973 Program) with No.2007CB307102 and National Hi-tech Research and Development Program of China (863 program) with No.2005AA121210. We thank

several members of Information Engineering Institute for their technical suggestions, including Peng Yi, Yufeng Li and Yang Li, and the anonymous reviewers for their constructive comments and suggestions.

References

1. Benko, P., Veres, A.: A passive method for estimating end-to-end tcp packet loss. In: Proceedings of IEEE Globecom (2002)
2. Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D.: Measurement and classification of out-of-sequence packets in a tier-1 ip backbone. In: Proceedings of Infocom 2003, pp. 113–114 (2003)
3. Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D.: Inferring TCP connection characteristics through passive measurements. In: Proceedings of Infocom 2004, pp. 1582–1592 (2004)
4. Jiang, H., Dovrolis, C.: Passive estimation of TCP round-trip times. ACM Computer Communication Review 32(3), 75–88 (2002)
5. Katti, S., Katabi, D., Blake, C., Kohler, E., Stauss, J.: M&M: A passive toolkit for measuring, tracking and correlating path characteristics. In: Proceedings of ACM Internet Measurements Conference (2004)
6. Zhang, Y., Breslau, L., Paxson, V., Shenker, S.: On the characteristics and origins of internet flow rates. In: Proceedings of ACM Sigcomm 2002, pp. 309–322 (2002)
7. Smith, M.A., Ramakrishnan, K.K.: Formal specification and verification of safety and performance of tcp selective acknowledgment. IEEE/ACM Trans. Netw. 10(2), 193–207 (2002)
8. Lee, D., Chen, D., Hao, R., Miller, R., Wu, J., Yin, X.: A formal approach for passive testing of protocol portions. In: IEEE International Conference on Network Protocols, pp. 122–131 (2002)
9. Cisco NetFlow, http://www.cisco.com/warp/public/732/Tech/netflow

The Design and Implementation of the DVS Based Dynamic Compiler for Power Reduction*

Xiang LingXiang [1], Huang JiangWei [1], Sheng Weihua[2], and Chen TianZhou[1]

[1] College of Computer Science, ZheJiang University, Hangzhou 310027, China
tzchen@zju.edu.cn
http://embedded.zju.edu.cn
[2] School of Electrical and Computer Engineering, Oklahoma State University,
202 Engineering South, Stillwater, OK, 74078
weihua.sheng@okstate.edu

Abstract. Recent years, as the wide deployment of embedded and mobile devices, reducing the power consumption in order to extend the battery life becomes a major factor that a designer must consider when designing a new architecture. DVS is regarded as one of the most effective power reduction techniques. This paper focuses on run-time compiler driven DVS for power reduction, especially two key design issues including DVS analysis model and DVS decision algorithm. Based on the design framework presented in this work, we also implement a run-time DVS compiler which is fine-grained, adaptive to the program's running environment without changing its behavior. The obtained system is deployed in a real hardware platform. Experimental results, based on some benchmarks, show that with average 5% performance loss, the benchmarks benefit with 26% dynamic power savings and the energy delay product (EDP) improvement is 22%.

Keywords: dynamic compiler, DVS, low power.

1 Introduction

Performance is always playing a major role in evaluating computer systems, but pursuing performance blindly, which normally leads to tremendous energy consumption, would cause disastrous limitation on development of computer systems [1]. This contradiction is particularly conspicuous when it comes to battery-powered embedded devices and mobile computers. Power-aware technologies can be grouped into two categories: hardware and software. Researches on power-saving of software focus on system software, which is normally closely related to system architecture, particularly operating systems and compilers.

Among all of CPU power saving techniques, Dynamic Voltage Scaling (DVS) [2] is a software controlled, runtime and dynamic technology that changes the supply voltage and clock frequency of the CPU automatically during the running of an application. When and which DVS strategy should be used to save the most energy while

* This work is supported by National Nature Science Foundation of China (60673149).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 233–240, 2007.

performance lose is limited in an acceptable range is the problem that DVS research tries to solve.

Dynamic compiler is a kind of system software which compiles, modifies and optimizes the binary code at runtime. It runs between the binary code of operating system and user applications. Compared to static compiler, dynamic compiler doesn't need any support from the source code. The feature of runtime complication leads to better understanding of the application's behavior and performance in real runtime environment. IBM DAISY [3], Intel PIN [4] and Valgrind [6] are typical dynamic compiler frameworks. Besides regular optimization, dynamic compiler can also be used to drive DVS for energy savings. Because dynamic compiler knows not only internal structure of a procedure, but also real runtime states, compared to DVS energy savings driven by operating system and static compiler, dynamic compiler is much more effective and competitive.

This paper discusses the model of DVS power-aware compiler, and implements a runtime DVS power-aware compiler under the framework proposed in [5]. By using the improved DVS decision algorithm, the immediate response of the DVS power-saving compiler is enhanced remarkably. Finally, some key issues of the design of this runtime DVS power-aware compiler are probed into deeply.

2 Design of the Dynamic Compiler for Power Reduction

2.1 Application Model

Several recent commercial processors have included DVS capabilities such as Transmeta Crusoe, Intel Xscale and Pentium-M. These processors' DVS setting is in the form of discrete frequency/voltage pairs. Changing the frequency means changing the voltage and vice versa. The DVS setting usually can be modified via privileged instructions.

According to the model proposed by Xie et al. [13], the CPU instructions can be classified into two categories corresponding to different operands: memory and computation operations. The operand of the former is the memory (in the case of cache hitting, the actual operand is cache). The latter's operand is register file. A normal code region (i.e., a loop, a function) is mixed of these two types of operations. As illustrated in Fig. 1, if more time spends on computation operations of a code region, the total executing time is limited to the CPU frequency. Otherwise, the executing time is limited to the memory speed (in this moment, the CPU idles for the memory operations). We call the former case as computation bounded code and the latter as memory bounded code.

For the memory bounded code region, the energy savings can be achieved via reducing the CPU frequency. As shown in Fig. 2a, the time of memory operations ($t_{invarient}$, which is independent of the CPU clock) in the block is much longer than the time of computation operations ($N_{dependent}/f$, in order to simplify the further calculation, the cache operating time N_{cache} has been added to $N_{dependent}/f$). The difference between them is called CPU slack time. In this case, as seen in Fig. 2b, if we slow the CPU (the new frequency is f'), the CPU slack time is eliminated. Although the application's execution time $t_{deadline}$ increases, on the whole the energy consumption due to increased execution time is still far less than the energy savings as the result of voltage and frequency slowdown.

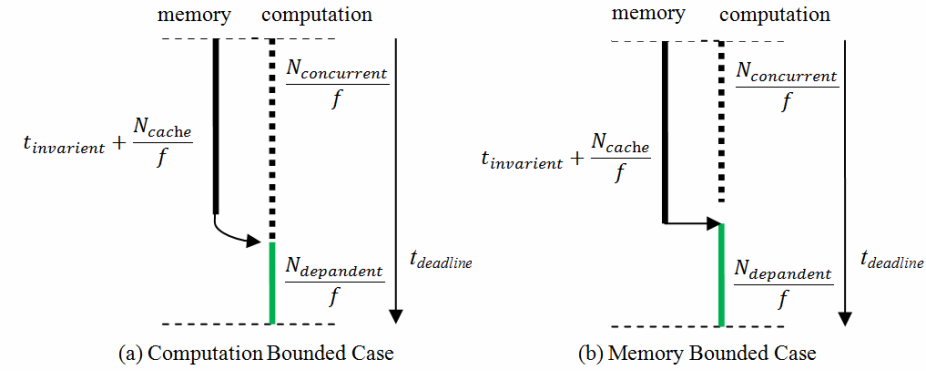


Fig. 1. Two Categories of Code Regions

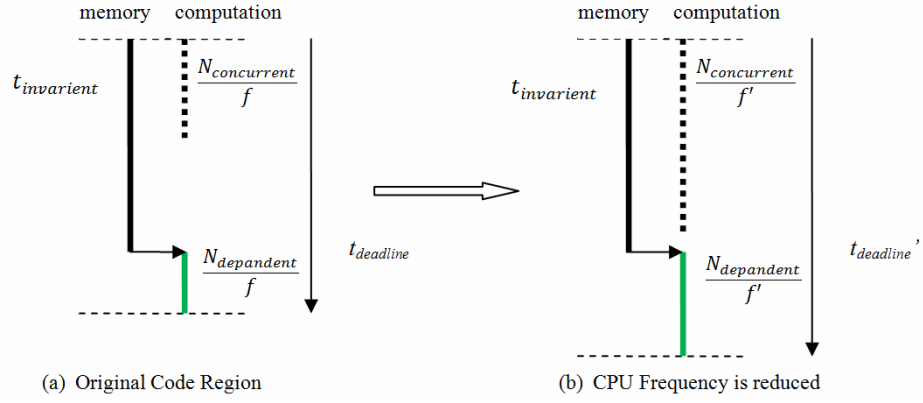


Fig. 2. Before and after the CPU frequency is reduced for the memory bounded region

2.2 DVS Analysis Model

We need a DVS algorithm to solve two problems:

1. When the performance degradation is restricted, it selects the suitable code regions to apply DVS. Performance degradation is caused by the delay of the entire application's execution time. There are a lot of researches of DVS [7, 8] concerning the real time application. This paper only considers the performance degradation of the entire application, ignoring the real time requirement.

2. Determinate an appropriate frequency/voltage setting for the selected code region to minimize the energy consumption.

In order to obtain the specific algorithm, this paper uses the DVS analysis model represented in [5]. Suppose the allowable maximum performance loss percentage is P_{loss}. We need to obtain the minimum frequency f' of a given candidate code segment, where $f'=\beta*f$. β is the ratio of new frequency of the candidate code segment to original frequency. We can obtain f' via calculating the corresponding β.

At the start, we denote the CPU slack time as:

$$\text{CPU slack time} = \mathrm{t}_{\text{invarient}} - \frac{N_{concurrent}}{totaltime} \tag{1}$$

Obviously, for any DVS candidate region, $\mathrm{t}_{\text{invarient}} > \frac{N_{concurrent}}{f}$, so CPU slack time > 0.

To estimate β, the equation 2 proposed in [5] is used:

$$\beta = 1 - P_{loss}\mathrm{k}_0 \times \frac{\mathrm{t}_{\text{invarient}}}{totaltime} + P_{loss}\mathrm{k}_0 \times \frac{N_{dependent}/f}{totaltime} \tag{2}$$

In this equation, k_0 is a constant relative to the hardware configuration. The value of $\frac{\mathrm{t}_{\text{invarient}}}{totaltime}$ and $\frac{N_{dependent}/f}{totaltime}$ are related to the detailed DVS algorithm.

This analysis model aims at a single code region. Ideally, all regions in the application should have specific performance loss P_{loss} to guarantee energy requirements. But in fact, for computation bounded code regions, there is no performance loss caused by DVS adjustment, so in summary, the overall performance loss of the entire application is actually less than P_{loss}. With this DVS analysis model, it is nearly impossible to make the entire program just at the point of the allowed maximum performance loss ratio, so as to utilize all the opportunities to save energy. But the cost is absolutely acceptable while considering there are few energy of this type can be saved.

This model ignores the extra energy consumption caused by DVS switch, which will make the analysis model complicated if we import it into this model. But the simplified model reduces calculation overhead and the energy consumption spending on DVS algorithm.

2.3 Design and Implementation

This section presents the runtime compiler's architecture. Then, two key design issues of power-aware dynamic compiler are discussed. The detailed DVS decision algorithm is given at last.

2.3.1 The Runtime DVS Compiler

We modify the dynamic compiler's architecture in [9] to satisfy the DBI engine and the DVS decision algorithm. The runtime compiler works between application's binary and operating system. It consists of three components, the code divider, the runtime monitor and the DVS optimizer.

Firstly, the code divider divides the original binary code into regions with fine grain and inserts profiling codes. At this moment, the application enters cold execution phase. In this stage, the runtime monitor finds the most frequently executed code regions (so-called *hot region*). Once the application enters in a hot region at first time, the DVS optimizer passes the hot region's runtime profiling data to DVS decision algorithm to check whether it is necessary to adjust the CPU frequency. If necessary, the DVS optimizer changes the DVS setting at latter running of this region.

2.3.2 Code Dividing

Similar to other dynamic optimizing technologies, we want only to analyze the application's most frequently executed part (hot region). There are numerable mythologies [10, 11, 12] to find the hot regions of an application. This paper identifies the hot code in a simply way. By dividing the binary code into several regions, we mark a code region as hot region when its execution number exceeds a threshold. A code region can be a single instruction, a block of instructions, branch, loop or function according to different grains. We choose the function as basic code region in order to be cost effective.

2.3.3 Profiling Code Placement

To collect the runtime performance data, the profiling code should be inserted in the application's binary. An ideal placement strategy can adjust the profiling code's location for accurate analysis. But in fact, we have to place the profiling code at fixed locations because of the uncertainty of an application's execution path and the limit of our DBI engine. In this paper, the profiling code is placed at before the start and after the termination of a function. Only the long run functions are profiled so the performance overhead is reduced further.

2.3.4 The DVS Decision Algorithm

From the DVS model above, we know that we must get the two values of $\frac{t_{invarient}}{totaltime}$ and $\frac{N_{dependent}/f}{totaltime}$ in equation 2 firstly in order to calculate the current frequency. It's unfortunate that no direct calculating means is provided by existing architectures. For the x86 platform, the optimizer in [5] (for convenience, we refer it as ParapetRDO) estimates the two values roughly using three PMCs of Pentium-M processor, BUS_TRAN_MEM, INST_RETIRED and UOPS_RETIRED. But the Pentium-M processor can only monitor two PMCs simultaneously, so we must run the application at least two times in order to obtain all three PMCs' value. Thus, this algorithm is non-immediate response.

This paper uses an improved algorithm. Based on the observation that the rate of INST_RETIRED and UOPS_RETIRED is very close to 1 for most code regions, we using the value of BUS_TRAN_MEM/UOPS_RETIRED to estimate a code region's memory boundedness directly. Experimental results show that the higher this rate is, the higher a code region's memory boundedness is. The improved algorithm is immediate response.

3 Evaluation and Experimental Results

A prototype of our power-aware compiler is deployed on a laptop with a Pentium-M processor (which provides 6 DVS settings ranging from 600MHz to 1700MHz). The OS is Linux kernel 2.6.20. The benchmarks are health[14], bw_mem[15] and mem (which mainly includes 7 functions. One of them is pure-calculating function. The other 6 functions have a mass of memory operations and have different memory behavior in terms of memory locality and memory boundedness). For evaluation, these benchmarks are compiled by gcc 4.1 with –O3. In order to reduce measurement error, we run each benchmark separately for three times and report the average results.

To eliminate the high overhead of the dynamic binary instrumentation engine (in this paper, we use Intel PIN with an basic dynamic energy overhead about 14%), the results reported here are based on the performance and dynamic energy value of each benchmark running with our dynamic compiler comparing to running with PIN. Table 1 summarizes the three benchmarks' statistical information collected by our dynamic compiler.

Table 1. Statistical information of benchmarks. Average L2 cache misses and Mem bus transections number are for per 1M upos.

Benchmark	Hot regions	DVS regions	Function name	Total Uops	Average L2 cache misses	Average Memory bus transections	DVS setting (MHz)
mem	7	4	fun_mem_l3	429M	21.8K	11.8K	1000
			fun_mem_l4	521M	52.2K	46.5K	600
			fun_mem_l5	520M	69.4K	54.4K	600
			fun_mem_l6	672M	80.3K	86.8K	600
bw_mem	14	1	rdwr	42M	157.1K	78.8K	600
health	4	0					

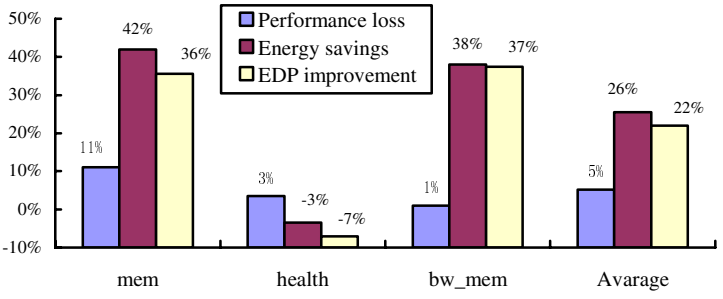


Fig. 3. Performance loss, energy savings and energy delay product (EDP) for benchmarks

As illustrated in the table, the statistical information matches with the benchmarks' program structure. But in some cases the dynamic compiler would choose a more suitable frequency setting if the processor provided more DVS settings.

Fig. 3 shows the normalized performance loss, dynamic power savings and energy delay product (EDP) improvement results for all benchmarks running with our dynamic compiler. It's obvious that the dynamic power-aware compiler works well for mem and bw_mem. For mem, the energy savings is 42%, the EDP improvement is 36% and the performance loss is 11%. For bw_mem, 38% energy savings and 37% EDP improvement are achieved with 1% performance loss. In contrast, for health, energy increases by 3% and EDP decreases by 7%. The contrast is caused by the reason that mem and bw_mem are high memory boundedness while health does not contain any DVS region (as seen in Table 1). So the overhead of PIN makes a negative effect. The results of health are significantly contrary to [5] (60% EDP improvement).

4 Conclusions

This paper has discussed the run-time compiler driven DVS for power reduction. Based on the ParapatRDO framework [5], we have designed and implemented a run-time power-aware compiler using the application model and the improved DVS decision algorithm described in this paper. The test on the improved DVS decision algorithm shows that it overcomes the non-immediate responsive defect in the ParapatRDO's algorithm and reduces the overhead of dynamic compiler. The experimental results show that with average 5% performance loss, the benchmarks benefit with 26% dynamic power savings and the energy delay product (EDP) improvement is 22%. The energy savings is up to 42% with 11% performance loss. This result indicates that run-time compiler driven DVS is an effective technology for power reduction.

References

[1] Venkatachalam, V., Franz, M.: Power reduction techniques for microprocessor systems. ACM Comput. Surv. 37(3), 195–237 (2005)
[2] Qu, G.: What is the limit of energy saving by dynamic voltage scaling. In: Proceedings of the International Conference on Computer Aided Design (2001)
[3] Ebcioglu, K., Altman, E.R.: DAISY: Dynamic compilation for 100% architectural compatibility. In: Proceedings of ISCA 1997 (June 1997)
[4] Luk, C.-K., Cohn, R., Muth, R., Muth, R., Patil, H., Kaluser, A., Lowney, G., Wallace, S., Reddi, V.J., Hazelwood, K.: PIN: Building customized program analysis tools with dynamic instrumentation. In: Proceedings of PLDI 2005 (June 2005)
[5] Wu, Q., Reddi, V.J., Wu, Y., Lee, J., Connors, D., Brooks, D., Martonosi, M., Clark, D.W.: A dynamic compilation framework for controlling microprocessor energy and performance. In: Proceedings of the 38th MICRO (November 2005)
[6] Nethercote, N., Seward, J.: Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In: Proceedings of PLDI 2007, San Diego, California, USA (June 2007)
[7] Dudani, A., Mueller, F., Zhu, Y.: Energy-Conserving Feedback EDF Scheduling for Embedded Systems with Real-Time Constraints. In: Proceedings of the joint Conference on Languages, Compilers and Tools for Embedded Systems, pp. 213–222. ACM Press, New York (2002)
[8] AbouGhazaleh, N., Mossé, D., Childers, B.R., Melhem, R.: Collaborative operating system and compiler power management for real-time applications. Trans. on Embedded Computing Sys. 5(1), 82–115 (2006)
[9] Wu, Q., Martonosi, M., Clark, D.W., Reddi, V.J., Connors, D., Wu, Y., Lee, J., Brooks, D.: Dynamic-Compiler-Driven Control for Microprocessor Energy and Performance. IEEE Micro 26(1), 119–129 (2006)
[10] Suganuma, T., Yasue, T., Nakatani, T.A: region-based compilation technique for dynamic compilers. ACM Trans. Program. Lang. Syst. 28(1), 134–174 (2006)
[11] Duesterwald, E., Bala, V.: Software profiling for hot path prediction: less is more. In: Proceedings of the Ninth international Conference on Architectural Support For Programming Languages and Operating Systems. ASPLOS-IX, pp. 202–211 (2000)

[12] Way, T., Breech, B., Pollock, L.: Region Formation Analysis with Demand-Driven Inlining for Region-Based Optimization. In: PACT 2000. Proceedings of the 2000 international Conference on Parallel Architectures and Compilation Techniques, vol. 24 (2000)
[13] Xie, F., Martonosi, M., Malik, S.: Compile-time dynamic voltage scaling settings: Opportunities and limits. In: Proc. of PLDI 2003 (June 2003)
[14] Carlisle, M.C., Rogers, A., Reppy, J.H., Hendren, L.J.: Early experiences with Olden. In: Proceedings of the 6th International Workshop on Languages and Compilers for Parallel Computing (August 1993)
[15] McVoy, L., Staelin, C.: lmbench: Portable Tools for Performance Analysis. Proceedings of USENIX 1996 Annual Technical Conference (1996)

Optimal Routing Algorithm and Diameter in Hexagonal Torus Networks*

Zhen Zhang [1,2], Wenjun Xiao[1], and Mingxin He[1,2]

[1] Dept. of Computer Science, South China University of Technology
Guangzhou, China 510641
[2] Dept. of Computer Science, Jinan University, Guangzhou, China 510632
zhang2003174@yahoo.com.cn,
wjxiao@scut.edu.cn, mx.he@yeah.net

Abstract. Nodes in the hexagonal mesh and torus network are placed at the vertices of a regular triangular tessellation, so that each node has up to six neighbors. The routing algorithm for the Hexagonal Torus is very complicated, and it is an open problem by now. Hexagonal mesh and torus are known to belong to the class of Cayley digraphs. In this paper, we use Cayley-formulations for the hexagonal torus, along with some result on subgraphs and Coset graphs, to develop the optimal routing algorithm for the Hexagonal Torus, and then we draw conclusions to the network diameter of the Hexagonal Torus.

1 Introduction

Hexagonal networks belong to the family of networks modeled by planar graphs. These networks are based on triangular plane tessellation, or the partition of a plane into equilateral triangles. The closest networks are those based on regular hexagonal, called honeycomb networks, and those based on regular square partitions, called mesh networks. Hexagonal networks and honeycomb have been studied in a variety of contexts. The Honeycomb architecture was proposed in [12], where a suitable addressing scheme together with routing and broadcasting algorithms were investigated, higher dimensional hexagonal networks have been defined in [5] and [4] as a generalization of the plane hexagonal networks. Addressing scheme, routing and broadcasting algorithms have been also proposed. An addressing scheme for the processors, and the corresponding routing and broadcasting algorithms for a hexagonal interconnection network has been proposed in [2]. The performance of hexagonal networks has been further studied in [3] and [11]. Hexagonal networks has been used in tracking mobile users and connection rerouting in Cellular networks[9]. The 2D hexagonal torus has been used in the HARTS project[13]. But the routing algorithm for the hexagonal torus has been an open problem.

* This research is supported by the Natural Science Foundation of Guangdong Province, China (No. 04020130).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 241–250, 2007.

Hexagonal mesh and torus, as well as honeycomb and certain other pruned torus networks, are known to belong to the class of Cayley graphs which are node symmetric and possess other interesting mathematical properties[15,16,17,19,7]. In this paper we use Cayley-graph formulations of hexagonal torus to develop an optimal routing algorithm, and then discuss the network diameter.

2 Knowledge of Cayley Graph

Before proceeding further, we introduce some definitions and notations related to digraphs Cayley digraphs in particular, and interconnection networks. For more definitions and mathematical results on graphs and groups we refer the reader to [1] and [6], for instance, and on interconnection networks to [8] and [10].A digraph $\Gamma=(V, E)$ is defined by a set V of vertices and a set E of directed edges. The set E is subset of elements (u, v) of If the subset E is symmetric, that is, $(u,v)\in E$, implies $(v,u)\in E$, we identify two opposite arcs (u,v) and (v,u) by the undirected edge (u,v). Let G be a group and S a subset of G. The subset S is said to be a generating set of G, and the elements of S are called generators of G, if every element of G can be expressed as a finite product of their powers. We also say that G is generated by S. The Cayley digraph of the group G and the subset S, denoted by $Cay(G,S)$, has vertices that are elements of G and arcs that are ordered pairs (g,gs) for $g\in E, s\in S$. If S is a generating set of G, we say that $Cay(G,S)$ is the Cayley digraph of G generated by S. When $1\notin S$ and $S=S^{-1}$, the graph $Cay(G,S)$ is a simple graph. Assume that Γ and Σ if for and $(u,v)\in E(\Gamma)$ we have $(\phi(u), \phi(v))\in E(\Sigma)$. In particular, if ϕ is bijection such that both ϕ and the inverse of ϕ are homeomorphisms, then ϕ is called an isomorphism of Γ to Σ. Let G be a group and S a subset of G. Assume that K is a subgroup of G, denoted as $K\leq G$. Let G/K denote the set of the right cosets of K in G. The right coset graph of G with respect to subgroup K and subset S, denoted by $Cos(G,K,S)$set of the right cosets of K in G. is the digraph with vertex set G/K such that there exists an edge (Kg, Kg') if and only if there exists $s\in S$ and $Kgs=Kg'$.

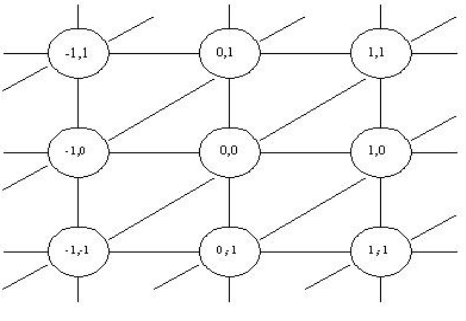


Fig. 1. Connectivity pattern for hexagonal mesh network

3 Hexagonal Mesh and Torus

3.1 Hexagonal Mesh

Let $G=Z\times Z$ where Z is the infinite cyclic group of integers, and consider $\Gamma=Cay(G,S)$ with $S=\{(\pm1,0),(0,\pm1),(1,1),(-1,-1)\}$. It is evident that Γ is isomorphic to the hexagonal mesh network [9][12]. Fig.1 shows a small part of an infinite hexagonal mesh in which the six neighbors of the "center" node (0,0) are depicted.

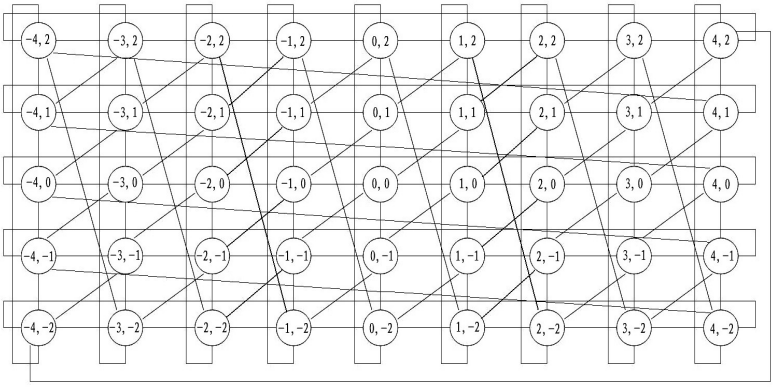


Fig. 2. Hexagonal torus with order 9×5

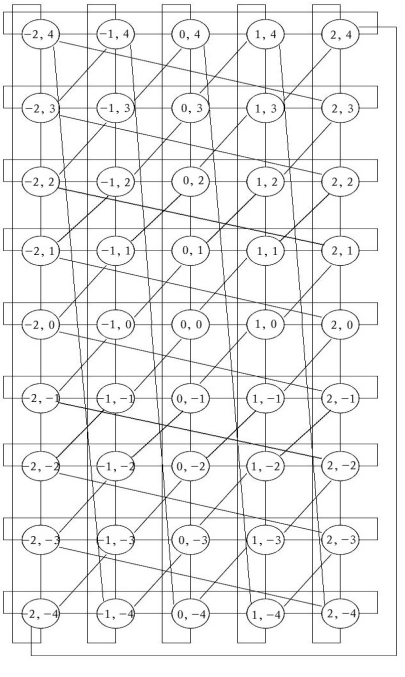


Fig. 3. Hexagonal torus with order 5×9

Using the Cayley-graph formulation of hexagonal networks, we can easily derive the distance $dis((a,b),(c,d))$ between the vertices (a,b) and (c,d) in such networks[18].

The routing algorithm of hexagonal mesh has been developed in [16] as the follow proposition.

Proposition 1. In the hexagonal mesh Γ, $dis((0,0),(a,b))$ equals $max(|a|,|b|)$ if a and b have the same sign and $|a|+|b|$ otherwise.

Proof. See [16].

By symmetry of Cayley graphs, we can easily obtain the distance between any two vertices in the graph Γ from Proposition 1, using $dis((a,b),(c,d))=dis((0,0),(c-a,d-b))$. This observation and the preceding discussion lead to a simple distributed routing algorithm for Γ.

3.2 Hexagonal Torus

Let $G=Z_l \times Z_k$, where Z_l and Z_k are cyclic groups of orders l and k respectively, $l>0,k>0$. Assume that S is defined as in the preceding paragraph. Then $\triangle=Cay(H,S)$ is the hexagonal torus of order lk. Fig.2 shows hexagonal torus with order 9×5 and Fig.3 shows hexagonal torus with order 5×9.

Using the results obtained for hexagonal meshes according to Proposition 1, we can deal with problems on Hexagonal torus which are, in general, more difficult. Let $\triangle$ be defined as above. Then we have the following result.

Proposition 2. For the hexagonal torus $\triangle$ of order lk and integers a and b, $l>a\geq 0,k>b\geq 0$,we have $dist((0,0),(a,b))=min(max(a,b),max(l-a,k-b),l-a+b,k+a-b)$.

According to the Proposition 2, we can develop a routing algorithm of the hexagonal torus.

4 Optimal Routing Algorithm for Hexagonal Torus

The hexagonal torus $\triangle=Cay(Z_l \times Z_k,S)$ with $S=\{(\pm 1,0),(0,\pm 1),(1,1),(-1,-1)\}$, is vertex transitive. The routing of any two nodes can transform to the routing of (0,0) to (a,b), a and b are integers. Let $a+ml \rightarrow a$ and $b+nk \rightarrow b$, with the choice of integer m and n, we can get

$$-\lfloor l/2 \rfloor \leq a \leq \lceil l/2 \rceil - 1 \qquad -\lfloor k/2 \rfloor \leq b \leq \lceil k/2 \rceil - 1$$

Now we discuss the routing from (0,0) to (a,b).

Definition 1. For any node (a,b) in the hexagonal torus, a is called the *x-dimension* coordinate and b is called the *y-dimension* coordinate. When $a+0 \rightarrow a$ or $a+1 \rightarrow a$ is called *x-dimension* increase, and when $x-0 \rightarrow x$ or $x-1 \rightarrow x$ is called *x-dimension* decrease. Also we can define *y-dimension* increase and decrease.

Proposition 3. The Optimal routing from (0,0) to (a,b) must keep *x-dimension* and *y -dimension* increase or decrease.

Proof. We only consider the case of first increase then decrease.

Case 1. *x-dimension* first increase then decrease.

Assume the routing from (0,0) to (a,b) is: (0,0) $\rightarrow$...$\rightarrow(a_i,b_j)$ $\rightarrow(a_i+1,b_k)$ $\rightarrow(a_i,b_m)$ $\rightarrow$…$\rightarrow(a,b)$, and $dis((0,0),(a,b))=D$. Because the generator $S=\{(\pm1,0),(0,\pm1),(1,1),(-1,-1)\}$, $b_k=b_j$ or b_j+1 and $b_m=b_k$ or b_k-1. We can get $b_m=b_j$ or b_j+1 or b_j-1, (a_i,b_m) may be (a_i,b_j) or (a_i, b_j+1) or (a_i, b_j-1).

If $(a_i,b_m)=(a_i,b_j)$, the (a_i,b_j) $\rightarrow(a_i+1,b_k)$ $\rightarrow(a_i,b_m)$ is a circle, the routing can change to (0,0) $\rightarrow$...$\rightarrow(a_i,b_j)$ $\rightarrow(a_i+1,b_k)$ $\rightarrow(a_i,b_m)$ $\rightarrow$…$\rightarrow(a,b)$, the distance change to D-2.

If $(a_i,b_m)=(a_i,b_j+1)$, the routing can change to (0,0) $\rightarrow$...$\rightarrow(a_i,b_j)$ $\rightarrow(a_i,b_j+1)$ $\rightarrow$…$\rightarrow(a,b)$, the distance change to D-1, this means the routing keeps *x-dimension* increase.

If $(a_i,b_m)=(a_i,b_j-1)$, the routing can change to (0,0) $\rightarrow$...$\rightarrow(a_i,b_j)$ $\rightarrow(a_i,b_j-1)$ $\rightarrow$…$\rightarrow(a,b)$, the distance change to D-1, this means the routing keep *x-dimension* decrease.

Case 2. The case of *y-dimension* first increase then decrease is similar to Case 1.

According to the Proposition 3, the routing between any two nodes can be divided into *x-dimension* routing and *y-dimension* routing. □

Definition 2. We define the functions $dist(\ )$ to denote the distance of two nodes, $dist_x(\ )$ to denote the *x-dimension* distance, $dist_y(\ )$ denote the *y- dimension* distance.

Now, we discuss the routing from (0,0) to (a,b) in five cases:

Case 1. One of a or b is zero
Case 2. $a>0$ and $b>0$.
Case 3. $a<0$ and $b<0$.
Case 4. $a>0$ and $b<0$.
Case 5. $a<0$ and $b>0$.

4.1 One of a or b is Zero

If $a=0$ and $b>0$, *x-dimension* and *y-dimension* keep increase to get the routing from (0,0) to (0,b): (0,0) $\rightarrow$(0,1) $\rightarrow$...$\rightarrow$(0,b).

If $a=0$ and $b<0$, *x-dimension* and *y-dimension* keep decrease to get the routing from (0,0) to (0,b) the routing from (0,0) to (0,b) keep *x-dimension* and *y-dimension* decrease : (0,0) $\rightarrow$(0,-1) $\rightarrow$...$\rightarrow$(0,b).

Similarly, we can get the routing in the case of $b=0$.

4.2 $a>0$ and $b>0$

For *x-dimension*, it has two ways to establish the routing from 0 to a: the increase way or the decrease way. According to the increase way, $dist_x((0,0),(a,b))=a$, while according to the decrease way, $dist_x((0,0),(a,b))=l-a$. Because $0 < a \le \lceil l/2 \rceil - 1$, we can get $\lfloor l/2 \rfloor + 1 \le l - a < l$, it is obviously that $l-a>a$. So the *x-dimension* routing must be increased. Because $b>0$, the *y-dimension* routing must be increased too. Synthetically, the routing between (0,0) to (a,b) must be *x-dimension* increased and *y-dimension* increased.

If $a \ge b$, the routing is:
(0,0) $\rightarrow$(1,1) $\rightarrow$...$\rightarrow(b,b)$ $\rightarrow(b+1,b)$ $\rightarrow$...$\rightarrow(a-1,b)$ $\rightarrow(a,b)$.
If $a<b$, the routing is:
(0,0) $\rightarrow$(1,1) $\rightarrow$...$\rightarrow(a,a)$ $\rightarrow(a,a+1)$ $\rightarrow$...$\rightarrow(a,b-1)$ $\rightarrow(a,b)$.

4.3 $a<0$ and $b<0$

The discussion of this part is similar to part 3.2, we can draw the conclusion that the routing between (0,0) to (a,b) is *x-dimension* decrease and *y-dimension* decrease.

If $|a|\geq|b|$, the routing is:

$(0,0) \rightarrow(-1,-1) \rightarrow ...\rightarrow(b,b) \rightarrow(b\text{-}1,b) \rightarrow ...\rightarrow(a+1,b) \rightarrow(a,b)$.

If $|a|<|b|$, the routing is:

$(0,0) \rightarrow(-1,-1) \rightarrow ...\rightarrow(a,a) \rightarrow(a\text{-}1,a) \rightarrow ...\rightarrow(a,b+1) \rightarrow(a,b)$.

4.4 $a>0$ and $b<0$

Because $-\lfloor k/2 \rfloor \leq b < 0$, we can get $k > k + b \geq k - \lfloor k/2 \rfloor = \lceil k/2 \rceil$, it is obviously that $k+b>|b|$. We have already known that $l-a>a$.

The routing between (0,0) to (a,b) can be implemented in four ways:

1. *x-dimension* increase and *y-dimension* decrease, the distance is $a+|b|$.
2. *x-dimension* increase and *y-dimension* increase, the distance is $max(a, k+b)$.
3. *x-dimension* decrease and *y-dimension* decrease, the distance is $max(l-a, -b)$.
4. *x-dimension* decrease and *y-dimension* increase, the distance is $l-a+k+b$.

According to the Proposition 2, we know that $dist((0,0),(a,b))=min(a+|b|, max(a, k+b), max(l-a, -b), l-a+k+b)$, then we discuss the routing problem in two cases:

(1) *$l-a>k+b$*

Because $l-a>k+b$ and $k+b>|b|$, we can get $max(l-a,-b)=l-a$, $l-a+k+b>l-a$, and $l-a>max(a, k+b)$, then $dist((0,0),(a,b))=min(a+|b|, max(a, k+b))$.

If $a\geq k+b$, $dist((0,0),(a,b))=min(a+|b|, a)=a$. So the routing is *x-dimension* increase and *y-dimension* increase:

$$(0,0) \rightarrow (1,1) \rightarrow ... \rightarrow (\lceil k/2 \rceil - 1, \lceil k/2 \rceil - 1) \rightarrow (k/2, -\lfloor k/2 \rfloor) \rightarrow (k/2+1, -\lfloor k/2 \rfloor + 1)$$
$$\rightarrow ... \rightarrow (k+b, b) \rightarrow (k+b+1, b) \rightarrow ... \rightarrow (a, b)$$

Example 1. The routing from (0,0) to (4,-2) in Fig.2 is as follows:

(0,0) →(1,1) →(2,2)→(3,-2) →(4,-2).

If $a<k+b$, $dist((0,0),(a,b))=min(a+|b|, k+b)$.

When $a+|b|\geq k+b$, , that is $a\geq k+2b$, we have $dist((0,0),(a,b))=k+b$. So the routing is *x-dimension* increase and *y-dimension* increase:

$$(0,0) \rightarrow (1,1) \rightarrow ... \rightarrow (a,a) \rightarrow (a,a+1) \rightarrow ... \rightarrow$$
$$(a, \lfloor k/2 \rfloor - 1) \rightarrow (a, -\lfloor k/2 \rfloor) \rightarrow (a, -k/2+1) \rightarrow ... \rightarrow (a,b)$$

Example 2. The routing from (0,0) to (1,-2) in Fig.2 is as follows:

(0,0) →(1,1) →(1,2)→(1,-2)

When $a+|b|<k+b$, that is $a<k+2b$, we have $dist((0,0),(a,b))=a+|b|$. So the routing is *x-dimension* increase and *y-dimension* decrease:

(0,0) →(1,0) →…→(*a*,0)→(*a*,-1) →…→(*a*,*b*) .

Example 3. The routing from (0,0) to (2,-1) in Fig.2 is as follows:

(0,0) →(1,0) →(2,0)→(2,-1) .

(2) *l-a≤k+b*

Because $k+b>l-a>a$, $max(a, k+b)=k+b$, $l-a+k+b>k+b$, and $k+b\geq max(l-a,|b|)$, we get $dist((0,0),(a,b))=min(a+|b|, max(l-a, |b|))$.

If $l-a\geq -b$, $dist((0,0),(a,b))=min(a+|b|, l-a)$.

When $a+|b|\geq l-a$, that is $b\leq 2a-l$, we have $dist((0,0),(a,b))=l-a$. So the routing is *x-dimension* decrease and *y-dimension* decrease:

$$(0,0)\rightarrow(-1,-1)\rightarrow ...\rightarrow(b,b)\rightarrow(b-1,b)\rightarrow ...\rightarrow(-\lfloor l/2\rfloor,b)\rightarrow$$
$$(\lceil l/2\rceil-1,b)\rightarrow(\lceil l/2\rceil-2,b)\rightarrow ...\rightarrow(a,b).$$

Example 4. The routing from (0,0) to (2,-3) in Fig.3 is as follows:

(0,0) →(-1,-1) →(-2,-2)→(2,-3).

When $a+|b|<l-a$ that is $b>2a-l$, we have $dist((0,0),(a,b))=a+|b|$. So the routing is *x-dimension* increase and *y-dimension* decrease:

(0,0) →(1,0) →…→(*a*,0)→(*a*,-1) →…→(*a*,*b*).

Example 5. The routing from (0,0) to (1,-2) in Fig.3 is as follows:

(0,0) →(1,0) →(1,-1)→(1,-2).

If $l-a<-b$, $k+b>-b>l-a$, we have $dist((0,0),(a,b))=min(a+|b|, |b|)=|b|$. So the routing is *x-dimension* decrease and *y-dimension* decrease:

$$(0,0)\rightarrow(-1,-1)\rightarrow ...\rightarrow(-\lfloor l/2\rfloor,-\lfloor l/2\rfloor)\rightarrow(\lceil l/2\rceil-1,-\lfloor l/2\rfloor-1)\rightarrow$$
$$(\lceil l/2\rceil-2,-\lfloor l/2\rfloor-2)\rightarrow ...\rightarrow(a,a-l)\rightarrow(a,a-l-1)\rightarrow ...\rightarrow(a,b).$$

Example 6. The routing from (0,0) to (2,-4) in Fig.3 is as follows:

(0,0) →(-1,-1) →(-2,-2)→(2,-3) →(2,-4).

4.5 a<0 and b>0

Because $0<b\leq\lceil k/2\rceil-1$ and $-\lfloor l/2\rfloor\leq a<0$, similar to part 4.5 we can get $k-b\geq b$ and $l+a\geq|a|$.

The routing between (0,0) to (*a*,*b*) can be implemented in four ways:

1. *x-dimension* decrease and *y-dimension* increase, the distance is $|a|+b$.
2. *x-dimension* increase and *y-dimension* increase, the distance is $max(l+a, b)$.
3. *x-dimension* increase and *y-dimension* decrease, the distance is $l+a+k-b$.
4. *x-dimension* decrease and *y-dimension* decrease, the distance is $max(|a|, k-b)$.

We know that $dist((0,0),(a,b))=min(|a|+b, max(l+a, b), l+a+k-b, max(|a|, k-b))$, then we discuss the routing problem in two cases:

(1) *k-b>l+a*

If $b\geq l+a$, $dist((0,0),(a,b))=min(|a|+b, b)=b$. So the routing is *x-dimension* increase and *y-dimension* increase:

$$(0,0)\rightarrow(1,1)\rightarrow ...\rightarrow(\lceil l/2\rceil-1,\lceil l/2\rceil-1)\rightarrow(-\lfloor l/2\rfloor,\lceil l/2\rceil)\rightarrow ...$$
$$\rightarrow(a,a+l)\rightarrow(a,a+l+1)\rightarrow ...\rightarrow(a,b).$$

Example 7. The routing from (0,0) to (-2,4) in Fig.3 is as follows:

(0,0) →(1,1) →(2,2)→(-2,3) →(-2,4).

If $b<l+a$, $dist((0,0),(a,b))=min(|a|+b, l+a)$.

When $|a|+b\geq l+a$, that is $b\geq l+2a$, we have $dist((0,0),(a,b))=l+a$. So the routing is *x-dimension* increase and *y-dimension* increase:

$(0,0) \rightarrow (1,1) \rightarrow ... \rightarrow (b,b) \rightarrow (b+1,b) \rightarrow ... \rightarrow (\lceil l/2 \rceil - 1, b) \rightarrow$
$(-\lfloor l/2 \rfloor, b) \rightarrow (-\lfloor l/2 \rfloor + 1, b) \rightarrow ... \rightarrow (a,b).$

Example 8. The routing from (0,0) to (-2,2) in Fig.3 is as follows:

(0,0) →(1,1) →(2,2)→(-2,2) .

When $|a|+b<l+a$, that is $b<l+2a$, we have $dist((0,0),(a,b))=b-a$. So the routing is *x-dimension* decrease and *y-dimension* increase:

(0,0) →(-1,0) →…→(a,0) →(a,1) →…→(a,b) .

Example 9. The routing from (0,0) to (-1,2) in Fig.3 is as follows:
(0,0) →(-1,0) →(-1,1)→(-1,2).

(2) *$k-b\leq l+a$*

Because $max(l+a,b)=l+a$, $l+a+k-b\geq l+a$, and $l+a\geq max(|a|,k-b)$, we get $dist((0,0),(a,b))=min(|a|+b, max(|a|, k-b))$.

If $k-b\geq -a$, $dist((0,0),(a,b))=min(a+|b|, k-b)$.

When $b-a\geq k-b$, that is $a\leq 2b-k$, we have $dist((0,0),(a,b))=k-b$. So the routing is *x-dimension* decrease and *y-dimension* decrease:

$(0,0) \rightarrow (-1,-1) \rightarrow ... \rightarrow (a,a) \rightarrow (a,a-1) \rightarrow ... \rightarrow$
$(a,-\lfloor k/2 \rfloor) \rightarrow (a,\lceil k/2 \rceil - 1) \rightarrow (a,\lceil k/2 \rceil - 2) \rightarrow ... \rightarrow (a,b).$

Example 10. The routing from (0,0) to (-2,2) in Fig.2 is as follows:

(0,0) →(-1,-1) →(-2,-2)→(-2,2).

When $b-a<k-b$, that is $a>2b-k$, we have $dist((0,0),(a,b))=b-a$. So the routing is *x-dimension* decrease and *y-dimension* increase:

(0,0) →(-1,1) →…→(a,0) →(a,1) →…→(a,b).

Example 11. The routing from (0,0) to (-2,1) in Fig.2 is as follows:

(0,0) →(-1,0) →(-2,0) →(-2,1).

If $k-b<-a$, $dist((0,0),(a,b))=min(a+|b|, |a|)=|a|$. So the routing is *x-dimension* decrease and *y-dimension* decrease:

$(0,0) \rightarrow (-1,-1) \rightarrow ... \rightarrow (-\lfloor k/2 \rfloor, -\lfloor k/2 \rfloor) \rightarrow (-\lfloor k/2 \rfloor - 1, \lceil k/2 \rceil - 1)$
$\rightarrow (b-k,b) \rightarrow (b-k-1,b) \rightarrow ... \rightarrow (a,b).$

Example 12. The routing from (0,0) to (-4,2) in Fig.2 is as follows:

(0,0) →(-1,-1) →(-2,-2)→(-3,2) →(-4,2).

The routing algorithm can get the optimal routing according to the Proposition 3.

5 Diameter of Hexagonal Torus

For any digraph Γ, $D(\Gamma)$ denotes the diameter of Γ, defined as the longest distance between any pair of vertices in Γ. In **[17]** ,we have the following result about the diameter.

Theorem 1. For $g \in S, S \subseteq G$, ,the mapping $\phi: g \to Kg$ is a homomorphism from $Cay(G,S)$ to $Cos(G,K,S)$.

Theorem 2. Assume that G is a finite group, $K \leq G, \Gamma = Cay(G,S)$, $\triangle = Cos(G,K,S)$ for some generating set S of G, and $D(\Gamma_K)$ denote the longest distance between vertices of K in Γ. Then we have $D(\Gamma) \leqslant D(\triangle) + D(\Gamma_K)$.
Proof. See [17].

Proposition 4. Assume the hexagonal torus $\triangle = Cay(Z_l \times Z_k, S)$,where $S=\{(\pm 1,0),(0,\pm 1),(1,1),(-1,-1)\}$, we have $\lceil (\max(l,k))/2 \rceil \leq D(\Delta) \leq \lfloor l/2 \rfloor + \lfloor k/2 \rfloor$.

Proof. Let $K=\{ (-\lfloor k/2 \rfloor,0) , (-\lfloor k/2 \rfloor+1,0) ,\ldots,(0,0),(1,0),\ldots, (\lceil k/2 \rceil-1,0) \}$, then $Cos(G,K,S)$ is an 1-D torus according to Theorem1and, $D(Cos(G,K,S))= \lfloor l/2 \rfloor$. The K in $\triangle$ is an 1-D torus too, and $D(\Gamma_K)= \lfloor k/2 \rfloor$. From the Theorem 2, we get $D(\triangle) \leqslant \lfloor l/2 \rfloor + \lfloor k/2 \rfloor$.For any $l \geq 3, k \geq 3$, either $dis((0,0),(1,\lceil k/2 \rceil)$or $dis((0,0),(\lceil l/2 \rceil,1))$ is $\lceil (\max(l,k))/2 \rceil$. Assume $k \geq l$, we can get $D(\triangle) \geq \lceil (\max(l,k))/2 \rceil$ according to the Proposion 2. □

6 Conclusion

The routing algorithm for the Hexagonal Torus is an open problem. In this paper, we use Cayley-formulations for the hexagonal torus to develop an optimal routing algorithm for the Hexagonal Torus. Then we discuss the diameter of the Hexagonal Torus, and give an upper bound and a lower bound to the diameter. We are currently investigating the Hexagonal Torus diameter in order to give an accurate value.

References

1. Biggs, N.: Algebraic Graph Theory. Cambridge University Press, Cambridge (1993)
2. Chen, M.S., Shin, K.G., Kandlur, D.D.: Addressing, Routing and Broadcasting in Hexagonal Mesh Multiprocessors. IEEE Trans. Computers 39(1), 10–18 (1990)
3. Dolter, J.W., Ramanathan, P., Shin, K.G.: Performance Analysis of Virtual Cut-Through Switching in HARTS: A Hexagonal Mesh Multicomputer. IEEE Trans. Computers 40(6), 669–680 (1991)
4. Decayeux, C., Seme, D.: 3D hexagonal network: modeling, topological properties, addressing scheme, and optimal routing algorithm. IEEE Trans. on Parallel and Distrib. Sys. 16(9), 875–884 (2005)
5. García, F., Solano, J., Stojmenovic, I., Stojmenovic, M.: Higher dimensional hexagonal networks. Journal of Parallel and Distributed Computing 63(11), 1164–1172 (2003)

6. Heydemann, M.: Cayley Graphs and Interconnection Networks. In: Graph Symmetry: Algebraic Methods and Applications, pp. 167–224 (1997)
7. He, M.X., Xiao, W.J.: A Unified Addressing Schema for Hexagonal and Honeycomb Networks with Isomorphic Cayley Graphs. In: IMSCCS 2006. Proc. of 1st Int. Multi-Symp. of Computer and Computational Sciences, Hangzhou, China, vol. 1, pp. 363–368 (2006)
8. Leighton, F.T.: Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypecubes. Morgan Kaufmann, San Francisco (1992)
9. Nocetti, F.G., Stojmenovic, I., Zhang, J.Y.: Addressing and Routing in Hexagonal Networks with Applications for Tracking Mobile Users and Connection Rerouting in Cellular Networks. IEEE Trans. Parallel and Distributed Systems 13(9), 963–971 (2002)
10. Parhami, B.: Introduction to Parallel Processing: Algorithms and Architectures, Plenum (1999)
11. Robic, B., Silc, J.: High performance Computing on a Honeycomb Arcbhitecture. In: Proc. Int. ACPC Parallel Computation Conf. (1993)
12. Stojmenovic, I.: Honeycomb Networks: Topological Properties and Communication Algorithms. IEEE Trans. Parallel and Distributed systems 8(10), 1036–1042 (1997)
13. Shin, K.G.: HARTS: A Distributed Real-Time Architectrue. Computer 24(5), 25–35 (1991)
14. Tosic, R., Masulovic, D., Stojmenovic, I., et al.: Enumeration of Polyhex Hydrocarbons up to h=17. J. Chemical Infromation and Computer Sciences 35, 181–187 (1995)
15. Xiao, W.J., Parhami, B.: Some Mathematical Properties of Cayley Digraphs with Applications to Interconnection Network Design. Int. J. Computer Mathematics 82, 521–528 (2005)
16. Xiao, W.J., Parhami, B.: Further Mathematical Properties of Cayley Digraphs Applied to Hexagonal and Honeycomb Meshes. Discrete Applied Mathematics (to appear, 2007)
17. Xiao, W.J., Parhami, B.: Hexagonal and Pruned Torus Networks as Cayley Graphs. In: Proc. International Conf. on Communications in Computing, Las Vegas, June 21-24, 2004, pp. 107–112 (2004)
18. Xiao, W.J., Parhami, B.: Structural Properties of Cayley Digraphs with Applications to Mesh and Pruned Torus Interconnection Networks. Int. J. of Computer and System Sciences, Special Issue on Network-Based Computing (to appear, 2007)
19. Xiao, W.J., Parhami, B.: A Group Consruction Method with Applications to Deriving Pruned Interconnection Networks. IEEE Trans. on Parallel and Distrib. Sys. 18(5), 637–643 (2007)

Implementation and Performance Evaluation of an Adaptable Failure Detector in iSCSI

Guang Yang, Jingli Zhou, and Gang Liu

College of Computer Science & Technology,
Huazhong University of Science & Technology,
Wuhan, 430074, China
yangchgang@gmail.com

Abstract. Unreliable failure detectors have been an important abstraction to build dependable communication applications over iSCSI systems subject to faults. In this paper, we propose a new implementation of a failure detector. This implementation is a variant of the heartbeat failure detector which is adaptable and can support scalable applications. In this implementation we dissociate two aspects: a basic estimation of the expected arrival date to provide a short detection time, and an adaptation of the quality of service according to application needs.

Keywords: Failure detectors, Adaptable, iSCSI systems.

1 Introduction

Failure detectors are well-known as a basic building block for fault-tolerant iSCSI systems. The best important factor of such systems is the stabilization. Failure detectors can provide the environment of stabilization and security. Failure detectors are used in a wide variety of fields, such as network communication protocols[1], group membership protocols[2,3], computer cluster management[4],etc. The main issue is that failure detectors encapsulate the indeterminism of distinguishing a very slow process from a crashed one, leaving agreement protocols free from timing issues.

We propose a new implementation of failure detector. This implementation is a variant of the heartbeat detector which is adaptable and can support scalable applications. Our algorithm is based on all-to-all communications where each Target periodically sends an "I am alive" message to all Initiators that connect it. To provide a short detection delay, we automatically adapt the failure detection time as a function of previous receptions of "I am alive"messages.

2 Unreliable Failure Detectors

In this section, we present a short description of failure detectors and some metrics to compare their performances. Failure detectors are characterized by two properties: completeness and accuracy. Completeness characterizes the failure

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 251–260, 2007.

detector capability of suspecting every incorrect Target permanently. Accuracy characterizes the failure detector capability of not suspecting correct Targets. Two kinds of completeness and four kinds of accuracy are defined in [5], which once combined yield eight classes of failure detectors.

Strong completeness: There is a time after which every Target that crashes is permanently suspected by every correct Initiator.

Eventual strong accuracy: There is a time after which correct Targets are not suspected by any correct Initiators.

In parallel, [6] proposes a set of metrics that can be used to specify the Quality of Service (QoS) of a failure detector. The QoS quantifies how fast a detector suspects a failure and how well it avoids false detection.

Detection time (T_D): T_D is the time that elapses from Target's crash to the time when Initiator starts suspecting Target permanently.

Mistake recurrence time (T_{MR}): This measures the time between two consecutive mistakes.

Mistake duration (T_M): This measures the time it takes the failure detector to correct a mistake.

3 Failure Detection Strategies

3.1 The Push Strategy

The Push period Δ_i: $\Delta_i$is the time between two emissions of an "I am alive" message.

The timeout delay Δ_{to}: Δ_{to} is the time between the last reception of an "I am alive" message from q and the time where p starts suspecting q, until an "I am alive" message from q is received.

The advantage of this approach is that the detection time is independent from the last heartbeat. This modification increases the accuracy because it avoids premature timeout and outperforms the failure detection time.

3.2 The Pull Strategy

The interrogation periodΔ_i : $\Delta_i$is the time between two emissions of an "Are you alive ?" message.

The timeout delay Δ_{to}: $\Delta_{to}$is the time between the emission of an "Are you alive ?" message by p to q, and the time where p starts suspecting q, until p receives an "I am alive" message from q.

3.3 System Model

We suppose an iSCSI system with Ω Initiators and Φ Targets. Ω Initiators and Φ Targets distributed over an unreliable wide-area network. We assume unbounded

communication delays and that the communication links can lose messages. In this iSCSI system, we define a receiving part of failure detector as a set of every Initiator module, fd_1 to fd_n, where fd_q is attached to an Initiator $q \in \Omega$. Every sending part of failure detector is attached to a Target $p \in \Phi$. Our FD implements a Push-style monitoring algorithm [8] that is based on requests and acknowledgement messages.

Each fd_q maintains a list of Targets $Suspect_q$ that are currently supposed of being crashed. Thus, we say that an Initiator $q \in \Omega$ suspects a Target $p \in \Phi$, at local time t, the Target p is in the list of suspected Targets maintained by fd_q. A fd_q in Initiator makes mistakes by incorrectly suspecting a Target. In this context, the suspicion resolution is based only on a timeout for some event, i.e., the detector is a timeout based failure detector. In addition, suspicions are not necessarily stable: if q suspects p at a given instant, it can later learn that the suspicion was incorrect. Target p is then removed by fd_q from its list of suspicious Targets.

4 Adaptation of the Delays

4.1 Arrival Date Estimation

The QoS of the detection depends on the Δ_i and Δ_{to} parameters. The timeout delay Δ_{to} is important because it determines the detection time. The estimation for Δ_i uses local information that each Initiator possesses. This information is limited to the observation of heartbeat message arrival dates and the interrogation period Δ_i. On the other hand the arrival time of heartbeat messages can be altered by the network load and the host load. In our solution the failure detector is structured into two layers. The first layer makes an accurate estimation to optimize the detection time. The second layer can modulate this detection time with respect to the needs in terms of QoS. In this part, we compare two methods: the first one is proposed in [6], the second method is the one we propose.

Chen's estimation. In [6], this technique estimates the arrival time for heartbeat messages (EA) and adds a constant safety margin.

Each process q considers the n most recent heartbeat messages, denoted m_1,m_2... m_n. Let A_1, A_2...A_n be their receipt times according to q's local clock. When at least n messages have been received, EA_{k+1} can be estimated by:

$$EA_{(k+1)} \approx \frac{1}{n}\left(\sum_{i=k-n}^{k} A_i - \Delta_i \times i\right) + (k+1) \times \Delta_i \tag{1}$$

The next timeout delay Δ_{to} (which expires at the next freshness point τ_{k+1}) is composed of EA and α the constant safety margin. EA represents the theoretical arrival date. The safety margin is added to avoid false detections caused by transmission delay or processor overload.

$$\tau_{(k+1)} = \alpha_{(k+1)} + EA_{(k+1)} \tag{2}$$

Our estimation. We estimate the arrival time with the Chen's method and we evaluate the safety margin dynamically. Chen's estimation of $EA_{(i+1)}$ supposes that we compute an average of n last arrival dates for each estimation, but we can transform it into a recursive equation: until Initiator receives at least n heartbeat messages from Target, the Initiator estimates the next arrival time by:

$$U_{(i+1)} = \frac{A_i}{i+1} \times \frac{k \times U_i}{i+1} \tag{3}$$

the arrival date average

$$EA_{(i+1)} = U_{(i+1)} + \frac{i+1}{2} \times \Delta_i \tag{4}$$

with $U_{(1)} = A_0$

And when Initiator has received more than n heartbeat messages, it uses:

$$EA_{(i+1)} = EA_i + \frac{1}{n}\left(A_i - A_{(i-n-1)}\right) \tag{5}$$

We suppose that the safety margin is not constant. We adapt the safety margin each time it receives a message. We employ the prediction error-based margin ($\alpha_{(i+1)}$) in this system. Consider that the network traffic presents a significant variation.The margin $\alpha_{(i+1)}$ will have its value adjusted to the capacity of the predictor to hit the next sample. As soon as the prediction error changes, the margin will be quickly recalculated to accommodate that error.

Similarly to Jacobson's estimation method [7], the margin $\alpha_{(i+1)}$ adapts its value each time the failure detector receives a message and the network load has varied, i.e., according to the error of the last estimation. Then, under a new message reception, at time t, a new margin value is computed by

$$\alpha_{(i+1)} = \alpha_i + c\left(|EA_i - A_i| - \alpha_i\right) \tag{6}$$

where c is the smoothing constant. In this paper, we set c=0.25 to obtain a fast reaction to error variation.

The next timeout $\Delta_{to(i+1)}$, activated by Initiator when it receives m_i, expires at the next freshness point:

$$\tau_{(k+1)} = EA_{(i+1)} + \alpha_{(i+1)} \tag{7}$$

4.2 Dynamic Adaptation of the Interrogation Delay

Failure detectors are designed to be used over long periods of time. The needs in terms of QoS are not constant, they vary according to each application. To adapt the QoS, we can change the interrogation delay Δ_i. The other reason for adapting the QoS of the detector is to adapt the bandwidth required by detectors with respect to the network load.

The idea is to allow the adaptation of Δ_i during the execution. In order to achieve this, all the detectors must reach a consensus over the new Δ_i.

The reasons to make this change are: the deliberate increase or decrease of the quality of detection, situations where the network capacity cannot allow to maintain the current quality of detection anymore, or where the network capacity increases and allows to obtain a higher quality of detection.

5 Failure Detector Algorithm

5.1 Algorithm

We implement a basic failure detection service, which provides an estimation for the arrival date of the next heartbeat message optimized with respect to detection time. This estimation is obtained from the expected arrival date and a dynamic margin. The aim of this layer is not to avoid all false detections but to provide a compromise between the number of false detections and the accuracy of the detection time. Algorithm of FD shows the whole algorithm of our failure detector.

Algorithm of FD:

1.Initialization:

1.1$Suspect_t \leftarrow \varnothing$

1.2 for all Target

1.3 $\Delta_I(T) = 0$ {$\Delta_I(T)$moderate the detection}

1.4 $\tau_0(T) = 0$ {Initially, all Target will be suspected by Initiator}

1.5 $EA_0(T) = U_0(T) = 0$, $delay_0(T) = initia\ value$,$\alpha_0 = 0$,$A_0(T) = 0$

1.6 $K(T) = -1$ {$K(T)$keeps the largest sequence number in all the messages Initiator received from Target so far}

2.Target:

2.1 at time $i \times \Delta_i$, sends heartbeat $m_i$to Initiator

3. Initiator:

3.1 upon receive message m_i at time t from Target

3.2 if $j > K(T)$ then $K(T) \leftarrow j$, $\alpha_{(i+1)} = \alpha_i + c(|EA_i - A_i| - \alpha_i)$

3.3 if $j < n$ then

3.4 $U_{(i+1)} = [i/(i+1)] \times [i/(i+1)]U_i$,$EA_{(i+1)} = U_{(i+1)} + [(i+1)/2]\Delta_i$

3.5 else $EA_{(i+1)} = EA_i + (t - A_{(i-n-1)}(T))/i$

3.6 end if

3.7 $A_i(T) \leftarrow t$ and $\tau_{(i+1)}(T) = EA_{(i+1)}(T) + \alpha_{(i+1)}(T)$

3.8 if Target $q \in Suspect_I$ then

3.9 $Suspect_I \leftarrow Suspect_I - q$ {trust q }

3.10 $State_q = S \rightarrow T$ and $\Delta_I(T) = \Delta_I(T) + 1$

3.11 end if

4. Task:

4.1 upon $\tau_{(i+1)}(T) = the\ current\ time$

4.2 wait during $\Delta_I(T)$ and if no message receive from Target q

4.3 $Suspect_I \leftarrow Suspect_I \bigcup q$

4.4 $State_q = T \rightarrow S$

The Initiator estimates the expected arrival date EA_i and the safety margin α for the next "I am alive" message from Target. From these results, Initiator

determines the next freshness point τ_i for Target. If Initiator currently suspects Target, then it stops suspecting it and increases its moderator timeout $\Delta_{I(t)}$ because Initiator knows that its previous timeout on Target was premature.

Task starts when Initiator does not receive an "I am alive" message from Target before the next freshness point $\tau(T)$. Initiator waits again for a message from Target during the moderator timeout $\Delta_I(T)$. If after this delay, Initiator still does not receive a message from Target, it starts suspecting it.

5.2 Proof

A failure detector must verify the two properties represented by the two Theorems 1 and 2.

Theorem 1(Strong completeness). Eventually every Target that crashes is permanently suspected by every correct Initiator.

$$\exists t_0 : \forall t \geq t_0, \forall p \in correct(I), \forall q \in crashed, q \in suspect_p(T) \tag{8}$$

Theorem 2(Eventual strong accuracy). There is a time after which the correct Targets are not suspected by every correct Initiator.

$$\exists t_{bound}, \forall t \geq t_{bound}, \forall p, q \in correct(I), q \notin suspect_p(T) \tag{9}$$

Strong Completeness

Theorem 1 is verified if Lemma 1 and 2 are verified. That is if there is a time t_{mute} after which no correct Initiator receives heartbeat messages from the crashed Target, and if there is a time $t_{timeout}$ after which all correct Initiators permanently suspect the Target.

Lemma 1. If Target crashes at t_{crash}, then there is a time t_{mute} after which Initiator stops receiving messages from Target.

$$t_{mute} \leq t_{crash} + \Delta_{msg} \tag{10}$$

Proof. We also assume that, at these instants, all the messages sent before the Global Stabilization Time (GST) have already been delivered and processed.

$$\exists t_{GST} : \forall m_i | t_{si} \geq t_{GST} : (t_{ri} - t_{si}) < \triangle_{msg}$$

$t_{si}$is the time when Target sends m_i and t_{ri} is the time when Initiator receives m_i

Suppose a Target crashed at t_{crash}. Then Target stops sending "I am alive" messages.

$$\nexists m_i | t_{si} \geq t_{crash}$$

The Initiator cannot receive message i from Target after $t_{ri} + \triangle_{msg}$. Hence Initiator cannot receive any message from Target after $t_{crash} + \triangle_{msg}$.

Lemma 2. For any sequence of i messages received by Initiator from Target, there is a time τ_i after which Initiator starts suspecting Target if it does not receive any message from Target.

Proof. From Algorithm of FD, when the Initiator receives a message $m_{(i-1)}$ from Target, it calculates a new τ_i after which it starts suspecting Target. If the τ_i is always bounded there is a time after which Initiator starts suspecting Target. We must prove that the τ_i is always bounded. The τ_i is calculated as follows:

$$\tau_i = EA_i + \alpha$$

If $i \leq n$

$EA_{(i+1)} = EA_i + \frac{1}{i}(A_i - A_0)$

.....

$EA_{(i+1)} = (A_1 - A_0) + \frac{1}{2}(A_2 - A_0 + ... + \frac{1}{k}(A_i - A_0))$

where A_i ,i are bounder. So EA_i is bounded.

If $i > n$

$EA_i = EA_{(i-1)} + \frac{1}{n}(A_{(i-1)} - A_{(i-n-1)})$

.....

$EA_i = EA_{(n-1)} + \frac{1}{n}[(A_{(i-1)} + A_{(i-2)} + ... + A_{(i-n)}) - (A_{(n-1)} + A_{(n-2)} + ... + A_0)]$

$EA_i < EA_{(n-1)} + \frac{1}{n}(nA_{(i-1)} - nA_0) = EA_{(n-1)} + A_{(i-1)} - A_0$

The time that heartbeat message arrive Initiator exceeds the time that Target sends heartbeat messages. $t_{send}(i)$ represents the arrival time of the I message, $t_{send}(0) < A_0$. From Lemma 1, we can find $A_{(i-1)} < t_{send}(i-1) + \Delta_{msg}$. We can calculate:

$EA_i < EA_{(n-1)} + A_{i-1} - t_{send}(0) < EA_{(n-1)} + t_{send}(i-1) + \Delta_{msg} - t_{send}(0)$

Let $\Delta_1, \Delta_2, ..., \Delta_{i-1}$ be a series of sending period between the first message and the m_i message.

$EA_i < EA_{(n-1)} + \Delta_{msg} + (\Delta_{(i-1)} + \Delta_{(i-2)} + ... + \Delta_1)$

We can find $EA_{(n-1)} = A_0 + \Delta_{(n-1)}$. So

$EA_i < EA_{(n-1)} + \Delta_{msg} + (i-1)\Delta_{max}$

Therefore EA_i is bounded. From [6], we can conclude: $\Delta_{(i+1)} + \alpha_{(i+1)} < T_D^U$, namely $0 < \alpha_i \leq T_D^U$. So α_i is bounded.

From Task, every time where Initiator times out and Target is correct then $\Delta_I(T)$ is increased. There is a time t_{bound} where $\Delta_I(T)$ is large enough to avoid false detection and stops increasing. When $\Delta_I(T)$ becomes upper than Δ_{msg} then no false detection can happened.

$\exists t_{bound}, \forall t \geq t_{bound}, \Delta_I(T) \geq \Delta_{msg} and \Delta_I(T)(t) = \Delta_I(T)(t+1)$

From Lemma 1 and Lemma 2, the strong completeness is proved. If for each message m_k received from Target, Initiator activates a bound timeout, then there is a time after which Initiator suspects Target, if it receives no new message from Target.

Eventual Strong Accuracy

The Theorem 2 is verified if the $\tau_i(T)$ of Initiator is large enough to avoid that Initiator wrongly suspects Target. From our model, if the lemma 3 is verified then the Theorem 2 is a direct deduction.

Lemma 3. There is a time after which τ_i is greater than $t_{si} + \Delta_{msg}$.

$$\exists t_{bound}, \forall t \geq t_{bound}, \tau_i \geq t_{si} + \tau_{msg} \tag{11}$$

Proof. From partial results 1, 2 and 3 we can say that:

$$\forall m_i \begin{cases} t_{si} < EA_i \\ 0 \leq \alpha_i \end{cases}$$
$$\exists t_{bound}, \forall t > t_{bound}, \tau_i > t_{si} + \Delta_{msg}$$

The Theorem 2 is verified because if τ_i is larger than $(t_{si} + \tau_{msg})$ then Target cannot be considered as having failed by Initiator.

6 Performances

6.1 Performance Evaluation

Fig 1 illustrates the evolution of the delay when the service starts. It presents a representative example from the detector on Initiator. Here the initial Δ_{to} is equal to 5700 ms. The detector must then adapt the margin to optimize the detection time.

The real delay between two receptions of heartbeat messages is nearly constant. In spite of this Chen's estimation is not a dynamic method so the Δ_{to} is almost constant. Our estimation is always higher than the Chen's estimation. This comes from the fact that our expected time evaluation is obtained with the real arrival dates average, and the real deviations are most often positive.

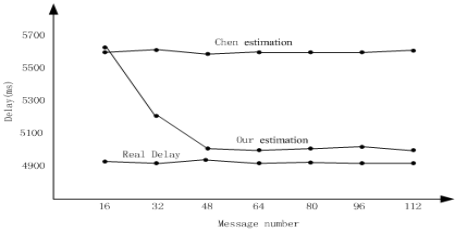


Fig. 1. Δ_{to} evolution at detector initialization

This experimentation shows that our estimation allows to avoid more false detections than Chen's estimation, and at the same time upholds a better detection time than Chen's estimation. This experimentation is summarized in Table 1.

Table 1. Summary of constant load experiment

	Our estimation	Chen's estimation
number of false detections	3	3
Mistake duration average (ms)	56	51
Detection Time average (ms)	5016	5089

This experimentation is in accordance with the previous result: our estimation is a compromise between a good detection time and the need to avoid false detections.

6.2 Evaluating the QoS

In this section we verify whether our algorithm can satisfy the QoS of failure detecting or not.

Table 2 shows the T_D is smaller than T_D^U when the system work on the condition that T_{MR}^L and T_M^U are same. We set T_M^U as 1500ms and $T^L MR$ as 10000ms.

Table 2. T_D(ms) in the different T_D^U

T_D^U	4000	4500	5000	5500	6000
T_D	2213	2567	2938	3306	3752

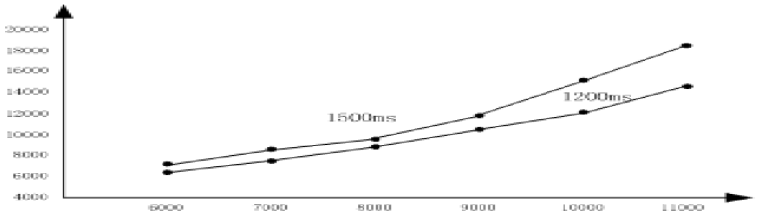


Fig. 2. Δ_{to} evolution at detector initialization

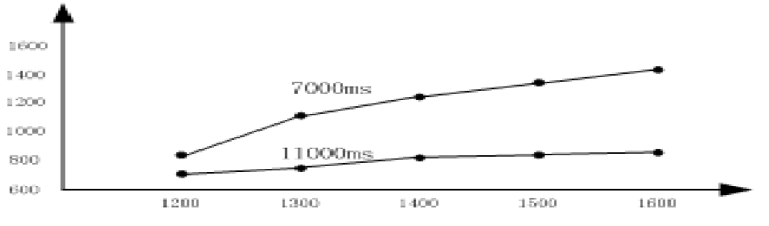


Fig. 3. Mistake duration

From Table 2, we can know the T_D is within the T_D^U .

Fig 2 illustrates that when the T_{MR}^L changes the Mistake recurrence time(T_{MR}) is larger than the T_{MR}^L on the condition that T_D^U and T_M^U are same . We set T_D^U as 5000ms, T_M^U as 1200ms and 1500ms. Fig 3 illustrates that when the T_{MR}^U changes the Mistake duration(T_M) is smaller than the T_{MR}^L on the condition that T_D^U and T_{MR}^L are same. We set T_D^U as 5000ms, T_{MR}^L as 7000ms and 10000ms.

From above results, our failure detector can satisfy the request of QoS.

7 Conclusion

In this paper, we have presented a new failure detector implementation in iSCSI. We dissociate two aspects: the first aspect, called the basic aspect, provides a basic estimation of the timeout delay Δ_{to} and the second aspect, called the Task, adapts the information provided by the first layer to the application needs. We have seen that our algorithm provides a good compromise between the optimization of the detection time and the need to avoid false detections.

The main characteristic of our implementation of the heartbeat failure detector whose are to be adaptive as well as dynamic, with a detection delay Δ_{to} composed of a short-term dynamic safety margin and a medium-term dynamic expected arrival date. This failure detector can also change its interrogation delay Δ_i to adapt its adequacy in terms of network load to the application needs and the network capacities.

References

1. Requirement for Internet Hosts-Communication Layers. In: Braden, R. (ed.) RFC 1122 (October 1989)
2. Amir, Y., Dolev, D., Kramer, S., Malkhi, D.: Transis: A Communications Sub-System for High Availability. In: Proc. 22nd Ann. Int'l Symp. Fault-Tolerant Computing, pp. 76–84 (July 1992)
3. Birman, K.P., van Renesse, R. (eds.): Reliable Distributed Computing with the Isis Toolkit. IEEE Computer Society Press, Los Alamitos (1993)
4. Pfister, G.F.: Search of Clusters, 2nd edn. Prentice Hall, Englewood Cliffs (1998)
5. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM (1996)
6. Chen, W., Toueg, S., Aguilera, M.K.: On the quality of service of failure detectors. In: Proc. of the First Int'l Conf. on Dependable Systems and Networks (2000)
7. Jacobson, V.: Congestion Avoidance and Control. In: ACM SIGCOMM 1988. Proceedings of the ACM Symposium on Communications, Architectures and Protocols, pp. 314–329. ACM Press, New York (1988)
8. Dolev, D., Friedman, R., Keidar, I., Malkhi, D.: Failure detectors in omission failure environments. In: Symp. on Principles of Distributed Computing, p. 286 (1997)
9. Nunes, R.C., Jansch-Poto, I.: QoS of Timeout-based Self-Tuned Failure Detectors: the Effects of the Communication Delay Predictor and the Safety Margin. In: DSN 2004. Proceedings of the, International Conference on Dependable Systems and Networks (2004)
10. Larrea, M., Fernandez, A., Arevalo, S.: Optimal implementation of the weakest failure detector for solving consensus. In: PODC 2000. Proc. of the 19th Annual ACM Symposium on Princi-ples of Distributed Computing, July 16-19, 2000, pp. 334–334. ACM Press, New York (2000)
11. Bertier MMarin OSens P.: Implementation and Performance Evaluation of an Adaptable Failure Detector. In: Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing (2004)
12. Shi, X.H., Jin, H., Han, Z.F., Qiang, W.Z., Wu, S., Zou, D.: ALTER: Adaptive failure detection services for grid. In: Cantarella, J.D. (ed.) Proc. of the IEEE Int'l Conf. on Services Computing, pp. 355–358. IEEE CS Press, Los Alamitos (2005)

A Niching Gene Expression Programming Algorithm Based on Parallel Model

Yishen Lin, Hong Peng, and Jia Wei

School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510641, China
Linnys@gmail.com, mahpeng@scut.edu.cn, wei.jia@mail.scut.edu.cn

Abstract. GEP is a biologically motivated machine learning technique used to solve complex multitude problems. Similar to other evolution algorithms, GEP is slow when dealing with a large number of population. Considering that the parallel GEP has great efficiency and the niching method can keep diversity in the process of exploring evolution, a niching GEP algorithm based on parallel model is presented and discussed in this paper. In this algorithm, dividing the population to the niche nodes in sub-populations can solves the same problem in less computation time than it would take on a single process. Experimental results on sequence induction, function finding and sunspot prediction demonstrate its advantages and show that the proposed method takes less computation time but with higher accuracy.

1 Introduction

Natural biological systems are well adapted to the environment; they can be used to solve many complex multitude problems. Inspired by the process of biological evolution in natural systems, evolutionary methods of algorithm designs are applied to stochastic searches for optimal results.

Gene Expression Programming (GEP) was first introduced by Candida Ferreira [1]. It combines the characteristics of Genetic Algorithms (GA) and Genetic Programming (GP), and overcomes some drawbacks of them. It has performed well for solving a large variety of problems, including symbolic regression, optimization, time series analysis, classification, logic synthesis and cellular automata, etc [1, 2, 3 and 4]. The GEP algorithm is a robust but slow process with a large number of individuals and complex multitude problems. Parallel execution is a better method to reduce computation time and to improve the efficiency in evolution algorithm. There are many studies in parallel GA [5, 6] and parallel GP [7, 8], but there are few studies in parallel GEP [9].

In this paper, a new algorithm called PNGEP (Parallel Niching GEP) which combines parallel model and niching method is presented. Experimental results on sequence induction, function finding and sunspot prediction show that this new algorithm gets better performance and higher efficiency than the basic GEP.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 261–270, 2007.

2 Related Works

Basic GEP can get good results in regression and prediction problem [1, 2, 3 and 4]. Niching method is a biologically technology, using this technology in evolution can get higher efficiency [10, 11 and 12]. However, similar to other evolution algorithms, GEP is also slow when dealing with a large number of individuals and complex multitude problems. To solve this problem, some researches have imported parallel model in evolution algorithm, and the hybrid algorithm has better performance [5, 6, 7, 8 and 9].

2.1 Niching Method

Niching method is widely used in GAs like Niching Genetic Algorithm (NGA). NGA are preserved the diversity inside the population by altering the operators to prevent premature convergence to an optimum result, like fitness sharing [10], crowding [11] and deterministic crowding [12] model.

For example, sharing fitness encourages individuals to populate proportionally over the whole search space by introducing a penalty on the fitness of each individuals based on its relative distance to its neighbors. This causes population diversity pressure that allows a population to maintain individuals at local optima, and reduce premature convergence [14]. This strategy will also force the final distribution of individuals to be dispersed throughout the niche. Each individual is under pressure to maximize distance between itself and its neighbors. This diversity pressure within the niches retards the exploration of the fitness peak areas in each niche, as fewer individuals are able to populate and explore the fitness peak areas.

2.2 Parallel Model

There are two parallel models in evolution algorithm: the coarse-grain model and the fine-grain model. In the coarse-grain model, the parallel program, which consists of a few computing-intensive processes, has few communication demands, such as the Message Passing Interface (MPI) model. The fine-grain one is made up of a large number of processes with low computational requirements but high demands on the communication in order to coordinate all the processes. The former utilizes fewer processors with less communication than the latter.

In the GEP algorithm, the individuals must be exchanged from each population, and the population in different processed must be cooperated with others. It is obvious that the fine-grain model is appropriate for applications if considering the balance of computational speed and precision.

This fine-grain model in GEP algorithm is also called the cooperation model. The processes sometimes exchange information by allowing some individuals to migrate from one process to another according for optimization. A share individuals' pool will be set. This approach re-injects diversity into converging processes. Then, different processes will be tended to explore different parts of the search space. This parallel model is in figure 1.

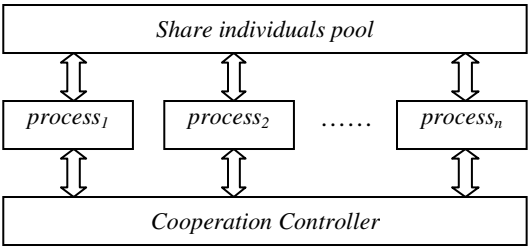


Fig. 1. Population is divided into several processes; the best individuals of each process will be exchanged through the share individuals' pool during the calculations. The cooperation controller controls the evolution of generation in each process.

3 Niching GEP Based on Parallel Model

In this paper, a hybrid algorithm called PNGEP is presented. This algorithm uses the fine-grain parallel model, which combines the niche theory and genetic mechanism.

3.1 Niching Method

The fundamental step of niching method is like the basic GEP. There is some different when the fitness of each individual is evaluated, a clustering of individuals operation will be done first. Before doing genetic operation, the individuals will be divided into k niches using the k-means clustering algorithm according to their fitness and *NMSE* value. The genetic operation will be done only in the same niche.

The main idea in this method is to define k centroids, one for each cluster. Each point is belonging to a given data set and associates it to the nearest centroid. Then recalculate k new centroids as new centers of the clusters resulting from the previous step, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. Finally k niches are set in a population with different types, such as good, average, poor, etc. Individuals only compete in the same niche and breed like in any traditional algorithm.

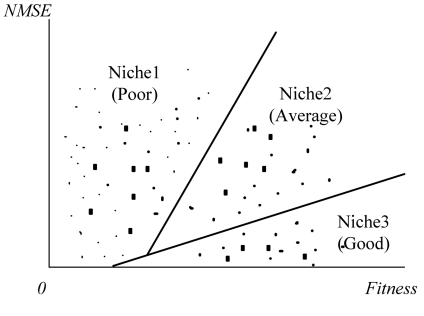


Fig. 2. First k centroids are defined; the individuals are selected and taken to the nearest centroid. Then re-calculate k new centroids of the clusters results and assign the individuals to the nearest new centroid. A loop has been generated. As a result of this loop the niche sets are initialization, the niches are marked like figure 2.

After doing the genetic operation in each niche, the *k* niches will compound to a new population and the elitism method will be used. This is one generation's operation, a loop will be generated. This clustering niching operation with *k*-means algorithm is shown in figure 2.

3.2 Parallel Model in Niching GEP

The main idea in this parallel algorithm is to define *N* sub-populations (processes), each sub-population with *k*-niche is mapped into a processor and its individuals are sometimes exchanged between the sub-populations during the calculations. The topology of this parallel model is shown in figure 3.

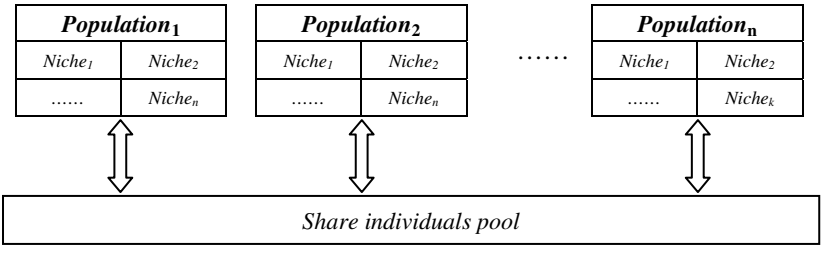


Fig. 3. Populations are divided into *N* sub-population and a sub-population is mapped into *k* niches. The best individuals of each sub-population will be exchanged through the share individuals' pool during the calculations.

In this parallel model, the best individual will be put into the share individual's pool and exchange to each sub-population. Then each sub-population will be re-injected the best genes. This behaves will be converged to a global/local optimum result.

3.3 Niching GEP Based on Parallel Model

PNGEP has seven genetic operators: mutation, transposition (insertion sequence transposition, root transposition and gene transposition), recombination (one-point, two-point and gene recombination). Among these operators, mutation is the most important and powerful one. PNGEP algorithm is depicted as follows:

```
Algorithm: PNGEP (Ts, Fs, f, P, Ps, k, N, G)

Input: Ts: the terminal set; Fs: the function set; f:
 the fitness function to evaluate the individuals; P: the
 sub-population for evaluation; Ps: the parameter for the
 genetic operation, such as the mutation rate, the multi-
 ple-point crossover rate, etc; k: number of the niches;
 N: number of the sub-populations; G: number of the gen-
 erations.

Output: The model with the highest fitness.
  1. For each sub-population:
       Initialize the sub-population P_i(i=1 to N) ran-
domly;
```

```
  2. For each generation g (g=1 to G)
      Evolution in each sub-population P_i (i=1 to N):
       (1)Inject: inject share-pool-individuals into P_i
random by pool exchange rate;
       (2)Evaluate: for each individual p, compute f(p);
       (3)Divide the individuals into k niches:
       (4)For each niche, generate the new population:

            (a) Mutation: generate new individual by mu-
            tation old individual.
            (b) Transposition: generate new individual by
            transposition old individual.
            (c) Recombination: generate new individual by
            recombination the two old individuals.

       (5) Using the elitism method;
       (6) Put the best m individual into share pool.

3. Return the best model with highest fitness.
```

4 Experiment and Results

In this paper, we compare PNGEP with the basic GEP in three problems [9, 15]. The first one is a problem of sequence induction, where a_n consists of the nonnegative integers. The n[th] term N of the chosen sequence is given by the formula:

$$N = 5a_n^4 + 4a_n^3 + 3a_n^2 + 2a_n + 1 \tag{1}$$

The second is a problem of "V" shaped function requiring floating-point constants. In this case, the following "V" shaped function is chosen:

$$y = 4.251a^2 + \ln(a^2) + 7.243e^a \tag{2}$$

where a is the independent variable and e is the irrational number 2.71828183.

Table 1. Wolfer sunspots series (read by rows)

101	82	66	35	31	7	20	92	154	125
85	68	38	23	10	24	83	132	131	118
90	67	60	47	41	21	16	6	4	7
14	34	45	43	48	42	28	10	8	2
0	1	5	12	14	35	46	41	30	24
16	7	4	2	8	17	36	50	62	67
71	48	28	8	13	57	122	138	103	86
63	37	24	11	15	40	62	98	124	96
66	64	54	39	21	7	4	23	55	94
96	77	59	44	47	30	16	7	37	74

The third one is the predicting sunspots problem. In this case, 100 observations of the Wolfer sunspots series are used (Table 1) with an embedding dimension of 10 and a delay time of one.

4.1 Setting the System

The relative error (equation 3), the absolute error (equation 4) and the normalized mean square error (*NMSE*, equation 5) are used to test the evaluation model.

$$fitness = \sum_{j=1}^{n} (M - |(y_j - y'_j)/y_j| * 100) \tag{3}$$

$$fitness = \sum_{j=1}^{n} (M - |y_j - y'_j|) \tag{4}$$

$$NMSE = \frac{\sum_{j=1}^{n} (y_j - y'_j)^2}{\sum_{j=1}^{n} (y_j - \bar{y}_j)^2} \tag{5}$$

In the equations, M is the range of selection; y_j is the fact value; $\bar{y}_j$ is the average of all y_j; y'_j is the value return by GEP. The less *NMSE* shows the good result.

For the sequence induction problem, the first 10 positive integers a_n are used as fitness cases. The fitness function is based on the relative error with a selection range of 20%, the maximum fitness is 200.

For the "V" shaped function problem, a set of 20 random fitness cases chosen from the interval [-1, 1] is used. The fitness function is also based on the relative error but in this case a selection range of 100% is used, the maximum fitness is 2000.

For the sunspot prediction problem, an embedding dimension of 10 and a delay time of one are used with 90 fitness cases. In this case, the fitness function is based on the absolute error with the selection range is 1000% and the maximum fitness is 90,000.

Because of the constants have less effect on the expected evolution; there is no constant using in the PNGEP algorithm. Our experiments show that the evolutionary results without constants of the three problems are good.

The PNGEP algorithm is written in C# using the threading class. *N* threads are created when the algorithm is initialized. Then *N* sub-populations are initialization and each sub-population is mapped into a thread process. When a-generation-running is done, the sub-populations exchange their individuals with the share stack. The best individuals will be re-injects diversity into converging sub-populations. Then, different sub-populations will be tended to explore different parts of the search space in this thread synchronization process.

In this paper, Experiments are running on a Hewlet-Packard BL25 blade server with AMD Opteron 265 1.8G CPU, 2G memory, Windows 2003 operation SP1 system and Microsoft .NET Framework 2.0 platform.

Table 2. General settings used in the sequence induction (SI), the "V" function and the sunspot prediction (SS) problems

	SI_{GEP}	SI_{PNGEP}	V_{GEP}	V_{PNGEP}	SS_{GEP}	SS_{PNGEP}
Number of runs	*100*	*100*	*100*	*100*	*100*	*100*
Number of generations	*100*	*100*	*200*	*200*	*200*	*200*
Population size	*200*	*50*	*200*	*50*	*200*	*50*
Niche number	*1*	*4*	*1*	*4*	*1*	*5*
Sub-population number	---	*2*	---	*4*	---	*4*
Number of fitness cases	*10*	*10*	*20*	*20*	*50*	*50*
Function set	{+,-,*,/}		{+,-,*,/, $ln,e^x,log,10^x,sin,cos$}		{+,-,*,/}	
Terminal set	{*a*}	{*a*}	{*a*}	{*a*}	{*a-j*}	{*a-j*}
Head length	*6*	*6*	*6*	*6*	*8*	*8*
Number of genes	*7*	*7*	*5*	*5*	*3*	*3*
Linking function	+	+	+	+	+	+
Chromosome length	*140*	*140*	*100*	*100*	*78*	*78*
Mutation rate	*0.044*	*0.044*	*0.044*	*0.044*	*0.044*	*0.044*
One-point recombination rate	*0.3*	*0.3*	*0.3*	*0.3*	*0.3*	*0.3*
Two-point recombination rate	*0.3*	*0.3*	*0.3*	*0.3*	*0.3*	*0.3*
Multipoint recombination rate	*0.3*	*0.3*	*0.3*	*0.3*	*0.3*	*0.3*
Gene recombination rate	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*
IS transposition rate	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*
IS element length	*1,2,3*	*1,2,3*	*1,2,3*	*1,2,3*	*1,2,3*	*1,2,3*
RIS transposition rate	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*	*0.1*
RIS element length	*1,2,3*	*1,2,3*	*1,2,3*	*1,2,3*	*1,2,3*	*1,2,3*
Selection range	*20%*	*20%*	*100%*	*100%*	*1000%*	*1000%*
Pool size	---	*4*	---	*8*	---	*8*
Pool exchange rate	---	*0.2*	---	*0.2*	---	*0.2*
Average best-of-run fitness	*151.674*	*184.370*	*1648.21*	*1780.04*	*88609.6*	*88643.1*
Average best-of-run *NMSE*	*0.0011*	*0.0005*	*0.0252*	*0.0132*	*0.3233*	*0.3108*
Average running time(second)	*<30*	*<30*	*572.52*	*106.34*	*215.46*	*84.73*
Success rate	*40%*	*71%*	---	---	---	---

4.2 Experimental Analysis

In the experiments, the selection is made by roulette-wheel sampling coupled with simple elitism and the performance is evaluated over 100 independent runs. The six experiments are summarized in Table 2.

The first problem of sequence induction can be exactly solved by the basic GEP and the PNGEP. The success rate of the basic GEP is 40% and the PNGEP is 71%. Both algorithms' running time is less than 30s. The PNGEP' precision is higher than the basic GEP.

To find the "V" shaped function, we use function set F = {+, -, *, /, *ln*, e^x, *log*, 10^x, *sin*, *cos*}. The basic GEP's average best fitness is 1648.21, the average best *NMSE* is 0.0252 and the running time is 572.52s. The PNGEP's average best fitness is 1780.04, the average best *NMSE* is 0.0132 and the running time is 106.34s. The basic GEP's running time is about five times longer than the PNGEP.

For the sunspot prediction problem, the basic GEP's average best fitness is 88609.6, the average best *NMSE* is 0.3233 and running time is 215.46s. The PNGEP's average best fitness is 88643.1, the average best *NMSE* is 0.3108 and the running time is 84.73s. The basic GEP's running time is about four times longer than the PNGEP. From the comparisons in table 2, we can see that the PNGEP is taken less computation time but with higher accuracy than the basic GEP.

The basic GEP often enters a local optimization and jumps out of the local optimization at random probability. On the other hand, PNGEP can jump out of the local optimization at a greater probability with *N* sub-population. For the sunspot prediction problem, the basic GEP search the solution space only with one population, but the PNGEP using the 4 sub-population. Although the basic GEP individuals' number is for times than the PNGEP, the PNGEP's individuals have more diversity by using the niche method. The comparison in figure 4 shows that PNGEP has better search ability than the basic GEP.

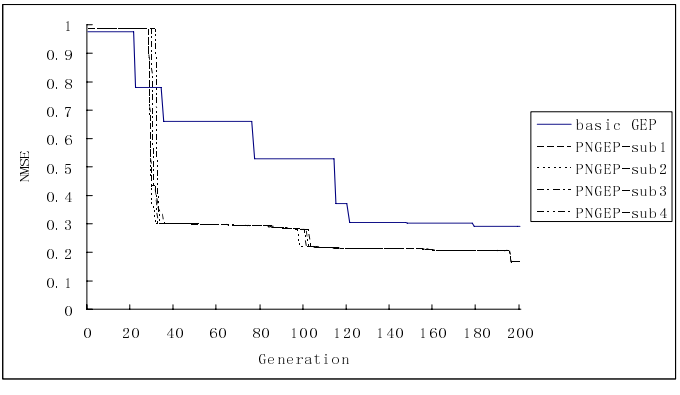


Fig. 4. The best solution's evolution in sunspot prediction problem between the basic GEP algorithm and the *4* sub-population PNGEP algorithm

Niching method tries to keep diversity in the population and to use this diversity as resource for exploratory evolution. The niche method of parallel model makes GEP with more flexibility and power of exploring the search space and converging to optimal result. From the comparisons of the success rate, the fitness value and the *NMSE* value, we can know that the PNGEP algorithm is better than the basic GEP.

5 Conclusion

In GEP algorithm, programs are represented as linear character strings of fixed-length which can be expressed as expression trees of different sizes and shapes. This separation of genotype and phenotype has endowed GEP with more flexibility and power of exploring the entire search space.

In this paper, a niching GEP based on parallel model is described and the advantages are demonstrated by its application. Experimental results on the sequence induction, the "V" shaped function and the sunspot prediction problem show that this parallel model of niching GEP algorithm, which called PNGEP, not only gains in the optimal results but also in better performance. It has higher precision and better search ability than the basic GEP. In the future, we will use the MPI parallel model and other clustering algorithm to improve the performance of this algorithm.

Acknowledgments

This research has been funded by the National Natural Science Foundation of Guangdong Province (07006474), Sci & Tech Research Project of Guangdong Province (2007B010200044) and Sci & Tech Research Project of Guangzhou (2006Z3-D3051).

References

1. Ferreira, C.: Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. Complex Systems 13, 87–129 (2001)
2. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence. Angra do Heroismo Portugal (2002)
3. Ferreira, C.: Automatically Defined Functions in Gene Expression Programming. Studies in Computational Intelligence 13, 21–56 (2006)
4. Jie, Z., Changjie, T., Chuan, L.: Time Series Prediction Based on Gene Expression Programming. In: Proceedings of the Fifth International Conference on Web-Age Information Management, Dalian, China (2004)
5. Gang, P., Iimura, I., Nakatsuru, T.: Efficiency of Local Genetic Algorithm in Parallel Processing. In: PDCAT 2005. Parallel and Distributed Computing, Applications and Technologies, pp. 620–623 (2005)
6. Goldberg, D.: Sizing population for serial and parallel genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, pp. 70-79 (1989)
7. Andre, D., Koza, J.R.: Parallel genetic programming: A scalable implementation using the transporter network architecture. In: Angeline, P., Kinnear, K. (eds.) Advances in Genetic Programming 2, Cambridge, MA, pp. 317–337 (1993)
8. Oussaidkne, M., Chopard, B., Pictet, O.: Parallel genetic programming and its application to trading model induction. Parallel Computing 23, 1183–1198 (1997)
9. Siwei, J., Zhihua, C., Dang, Z.: Parallel Gene Expression Programming Algorithm Based on Simulated Annealing Method. ACTA Electronic Sinica 33, 2017–2021 (2005)

10. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the 2nd International Conference on Genetic Algorithms, pp. 41–49 (1987)
11. De Jong, K.: An analysis of the behavior of a class of genetic algorithms. Dissertation Abstracts International 36(10), 5140B (1975)
12. Mahfoud, S.W.: Crowding and preselection revisited. Parallel Problem Solving from Nature II, 27–36 (1992)
13. Ferreira, C.: Gene Expression Programming and the Evolution of Computer Programs. In: Recent Developments in Biologically Inspired Computing, pp. 82–103. Idea Group Publishing (2004)
14. Yang, H., Ch, F., Li, C., Wang, M.: A density clustering based niching genetic algorithm for multimodal optimization. In: Machine Learning and Cybernetics. Proceedings of 2005 International Conference, vol. 3, pp. 1599–1604 (2005)
15. Ferreira, C.: Function Finding and the Creation of Numerical Constants in Gene Expression Programming. In: Proceedings of the 7th Online World Conference on Soft Computing in Industrial Applications (2002)

ComNET: A P2P Community Network

Zhentao Sun and Wenjun Xiao

Department of Computer Science, South China University of Technology, Guangzhou 510641, P.R. China
robinvane@163.com, wjxiao@scut.edu.cn

Abstract. In addition to searching, browsing is yet another requirement of P2P file sharing systems. Nevertheless, none of the recent P2P DHTs can closely connect peers with the same interests together so that it is not practical to provide browsing service in such systems. In this paper, we define a new cayley graph to support logical grouping, and based on this cayley graph, a set of P2P DHT protocols which is suitable for providing file browsing service is also designed. Performance evaluation indicates that the new protocols can reach the theoretical lower bound of routing table size and query path length. Furthermore, the robustness of ComNET is also better than most of the P2P DHTs recently proposed.

Keywords: P2P, ComNET, Grouping, Browsing service, Cayley graph, Small-world.

1 Introduction

Almost all recent researches of P2P DHT [1] [2] [3] concentrate on how to lower the length of query path and reduce the size of routing table. Therefore the users download behavior are not taken into consideration when they design the systems. In addition to searching, browsing is one of other important requirements when people use P2P file-sharing systems, but as we know, none of current P2P DHT can support efficient file browsing service.

To remedy the disadvantages of the recent structured P2P system, we introduce small-world phenomena [4] into the overlay network. The phenomena of small-world lead to the phenomena of community, which means that people with the same interests know each other with high probability (i.e. highly clustered). In order to introduce small-world features into P2P DHT, we define a new cayley graph Γ, and then based on Γ, a new P2P DHT called ComNET is also designed. In addition to efficient resource searching mechanism, ComNET supports explicit peer grouping, and thus supports effective resource browsing service. Both theoretical analysis and experimental evaluation show that ComNET can reach the lower bound[3] of routing table size and query path length at the same time. Moreover, its robustness is also better as compared to Ulysses[1].

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 271–281, 2007.

2 Related Research and Design Objectives

2.1 Related Research

The motivation for our research stems from the following fields:

1. Ulysses[1]. Ulysses is a P2P DHT based on the famous butterfly graph and can reach a diameter of $\mathcal{O}(\log n/\log\log n)$. Nevertheless, in order to cluster similar peers, Ulysses has to introduce another overlay which will cause performance issues.
2. Content-based shortcut Gnutella[6]. Built on Gnutella, this P2P network adds links between similar peers and removes rarely used links. However, this system is still unstructured which means that the overloading problem can not be solved and explicit peer grouping can not be provided in this system.
3. Cayley graph as models of small-world networks. W. Xiao and B. Parhami propose a model of deterministic small-world graph in [7]. They use a algebraic method to construct cayley graphs which display small-world features.

2.2 Design Objectives

We are primarily interested in the following features of P2P DHT network.

1. *Short query path*: The average length of query path would not increase significantly as the number of peers in the overlay network become large.
2. *Reasonable size of routing table*: The minimal routing table size is beneficial to ensure fault-tolerance while the maximal one is relevant for ensuring bounded maintenance cost[5].
3. *Reasonable cluster coefficient*(CC1)[4]: Non-zero cluster coefficient leads to the phenomena of clustering and community. Our resulting P2P system should have a reasonable CC1.
4. *Peer Grouping*: Performance of file sharing system can be remarkably improved by structured grouping, and with which, users can easily browse the files they are more interested in.
5. *Self-configuration*: It is not possible for a large scale P2P system to employ centralized server to provide joining, departing and searching services. For the sake of scalability, distributed network services are preferred.
6. *Robustness*: P2P systems should have the ability to deal with high dynamic environment so that its performance will not drop dramatically when there are faulty peers in the system.

3 The Definition of Static ComNET

Every cayley graph is vertex transitive, and thus using cayley graph as the static graph of P2P DHT has the benefit of distributing loading to all peers evenly[8]. The static graph of ComNET Γ is defined in this section and then some properties of this graph which are essential to P2P systems are explored.

3.1 Terminology and Notation

In this section, x and y are assumed to be strings composed of digital or asterisk "*". We define the following operations and predicates on them:

1. $|x|$: the length of string x
2. $x[i]$: the i^{th}(left to right, counting from index 0) character of x
3. $lock(x, y, i)$:$|x| \leq i \vee |y| \leq i \vee x[i] =$ "*" $\vee y[i] =$ "*" $\vee x[i] = y[i]$. For example, if $x =$ "210", $y =$ "2*11", then $lock(x, y, 0)$, $lock(x, y, 1)$, $lock(x, y, 3)$, but $\neg lock(x, y, 2)$
4. $lockall(x, y)$:$\forall j \in \mathbb{N}, lock(x, y, j)$
5. $AP(x, i, r)$[1], $i \in \mathbb{Z}_r$ and x is a binary string: A sub-string of x which is composed of $x[i], x[i+r], x[i+2r], \cdots$. For example $AP($"010*2", $1, 2)$ ="1*".
6. $APlock(x, y, i, r)$: defined as $lockall(AP(x, i, r), AP(y, i, r))$. For example, if x="010*1", y="110111", i=1, $r = 2$, then $AP(x, i, r) =$ "1*", $AP(y, i, r)$= "111", and thus $APlock(x, y, i, r)$. In this paper, $APlock(x, y, i, r)$ is referred to as *"x locks on y along dimension i"*.
7. $APlockset(x, y, r)$: is the maximal subset of $\mathbb{Z}_r$ which satisfies that $\forall l \in APlockset(x, y, r), APlock(x, y, l, r)$. This set is called *the locking set of x on y* or in brief, *the locking set of x* if the context is clear.
8. $APlockbut(x, y, i, r)$:$\forall j \in \mathbb{Z}_r \wedge j \neq i, APlock(x, y, j, r)$
9. $APlockall(x, y, r)$:$\forall j \in \mathbb{Z}_r, APlock(x, y, j, r)$
10. $m(\alpha, \beta, \gamma) : m(\alpha, \beta, \gamma)$ is obtained by replacing the γ^{th} character of α with the γ^{th} character of β.

3.2 The Definition of Cayley Graph Γ

Definition 1. *Let $H = (G, \bullet)$, where $G = (\mathbb{Z}_{r_c}^k, \mathbb{Z}_{r_p}^k, \mathbb{Z}_k)$. The operation $\bullet$ on G is defined as $\forall(\mathbf{c_1}, \mathbf{p_1}, r_1), (\mathbf{c_2}, \mathbf{p_2}, r_2) \in G$,*

$$(\mathbf{c_1}, \mathbf{p_1}, r_1) \bullet (\mathbf{c_2}, \mathbf{p_2}, r_2) = (\mathbf{c_1} \oplus \sigma^{r_1}(\mathbf{c_2}), \mathbf{p_1} \oplus \sigma^{r_1}(\mathbf{p_2}), r_1 + r_2)$$

σ is cyclic right shift operation. $\oplus$ is component-wise addition mod r_c and r_p. Unless noted otherwise, $+$ is modulo-k addition throughout this paper.

Corollary 1. *$(G, \bullet)$ is a group.*

For any $(\mathbf{c}, \mathbf{p}, r) \in G$, $\mathbf{c}$ is referred to as *group identifier*, $\mathbf{p}$ as *intra-group identifier*, r as *region identifier*, and $(\mathbf{c}, \mathbf{p}, r)$ as *vertex identifier*.

Group identifier make it possible for P2P DHT to provide peer grouping mechanism. Moreover, if the peers who share similar interests are clustered with the same group identifier, it would be more easier for them to reach each other, and therefore improve the efficiency of browsing operation.

Definition 2. *Let $S = S_p \cup S_c \cup S_r$, where $S_p = \{(\mathbf{0}, p0^{k-1}, 0) | p \in \mathbb{Z}_{r_p} \backslash 0\}$ $S_c = \{(c0^{k-1}, \mathbf{0}, 0) | c \in \mathbb{Z}_{r_c} \backslash 0\}$ $S_r = \{(\mathbf{0}, \mathbf{0}, r) | r \in \mathbb{Z}_k \backslash 0\}$, then $\Gamma = Cay(G, S)$ is a cayley graph.*

Links in Γ can be categorized to 3 types: $Link_p$, links between two vertices in the same group and region; $Link_c$, links between two vertices in different groups but in the same region; and $Link_r$, links between two vertices in different regions.

[1] From [1].

3.3 Some Properties of Γ

The degree of a cayley equals the cardinality of S, that is $|S| = |S_p|+|S_c|+|S_r| = r_p - 1 + r_c - 1 + k - 1 = r_p + r_c + k - 3$.

Proposition 1. *Γ is a $(r_p + r_c + k - 3)$-regular graph.*

The routing from $(\mathbf{c}, \mathbf{p}, r)$ to $(\mathbf{c_3}, \mathbf{p_3}, r_3)$ proceeds in two phases. In the first phase $\mathbf{c}$ and $\mathbf{p}$ successively change to $\mathbf{c}_3$ and $\mathbf{p}_3$. In the second phase, one step is required to correct the region identifier to r_3. The pseudo-code for forwarding in a vertex is shown in algorithm 1.

```
   Input  : Current vertex (c1, p1, r1) and destination vertex (c3, p3, r3)
   Output: Identifier of next-hop vertex (c2, p2, r2)
 1 if (c1 = c3 ∧ p1 = p3 ∧ r1 = r3) then
 2     the destination has been reached
 3 else
 4     if (c1 = c3 ∧ p1 = p3) then
 5         (c2, p2, r2) := (c3, p3, r3)
 6     else
 7         if (lock(c1, c3, r1) ∧ lock(p1, p3, r1)) then
 8             r2 is an integer that does not satisfy lock(c1, c3, r2) or
               lock(p1, p3, r2). Non-r3 integers are preferred;
 9             c2 := c1, p2 := p1
10         else
11             if (¬(lock(p1, p3, r1)) then
12                 p2 := m(p1, p3, r1), c2 := c1, r2 := r1
13             else
14                 c2 := m(c1, c3, r1), p2 := p1, r2 := r1
15             end
16         end
17     end
18 end
```

Algorithm 1. Routing algorithm at a vertex in Γ

We can obtain from the routing algorithm that,

Proposition 2. *The diameter of Γ is $2k + k = 3k$.*

Note that the number of vertices n of Γ is $k(r_c r_p)^k$. If we let $k = \log n / \log\log n$, then $r_c r_p = (n/\frac{\log n}{\log\log n})^{1/\frac{\log n}{\log n \log n}} < \log n$. According to proposition 1, the degree of a vertex is $r_p + r_c + k - 3 < \log n / r_c + r_c + \log n / \log\log n < \log n + 1 + \log n / \log\log n$, thus

Proposition 3. *The degree and diameter of Γ can reach a complexity of $\mathcal{O}(\log n)$ and $\mathcal{O}(\log n / \log\log n)$ respectively.*

Proposition 3 shows that Γ reaches the theoretical lower bounds proposed by paper [3].

Proposition 4. *CC1 of* Γ *equals* $(C^2_{r_c-1} + C^2_{r_p-1} + C^2_{k-1})/C^2_{r_c+r_p+k-3}$

We can see that Γ is highly clustered with CC1 $\gg CC1_{random\ graph}$, which shows the small-world phenomenon.

4 ComNET Protocols

Our P2P DHT uses Γ as its static graph. In this section, algorithms on how to embed peers into Γ to construct a P2P DHT overlay called ComNET are presented.

4.1 ComNET Basics

1. *The Identifier Space*: Every peer in ComNET is identified by a unique 3-tuple $(\mathbf{c}, \mathbf{p}, r)$:

$$\mathbf{c} \in \{(c_0 c_1 \cdots c_s) | -1 \leq s < l_c, c_i \in \mathbb{Z}_2 \cup \{*\}\}$$
$$\mathbf{p} \in \{(p_0 p_1 \cdots p_t) | -1 \leq t < \infty, p_i \in \mathbb{Z}_2\}$$
$$r \in \mathbb{Z}_k$$

   l_c and k are two integral parameters of ComNET.
2. *The Topology of ComNET*: The topology of ComNET captures the link structure of Γ. Geometrically, $(\mathbf{c_1}, \mathbf{p_1}, r_1)$ is adjacent to $(\mathbf{c_2}, \mathbf{p_2}, r_2)$ *if and only if*:
   (a) $r_1 = r_2 \wedge APlockall(\mathbf{c}_1, \mathbf{c}_2, k) \wedge APlockbut(\mathbf{p}_1, \mathbf{p}_2, r_1, k)$, link between them corresponds to $Link_p$ in Γ
   (b) $r_1 = r_2 \wedge APlockall(\mathbf{p}_1, \mathbf{p}_2, k) \wedge APlockbut(\mathbf{c}_1, \mathbf{c}_2, r_1, k)$, link between them corresponds to $Link_c$ in Γ
   (c) $APlockall(\mathbf{c}_1, \mathbf{c}_2, k) \wedge APlockall(\mathbf{p}_1, \mathbf{p}_2, k)$, link between them corresponds to $Link_r$ in Γ

   Figure 1 shows a ComNET composed of 10 peers
3. *Distribute the Hash Table*: File names in ComNET are hashed to 3-tuples (α, β, γ), where $\gamma \in \mathbb{Z}_k$, α and β are fixed strings which satisfy $|\alpha| = l_c$ and $|\beta| \gg |\mathbf{p}|$. For any key (α, β, γ) and peer identifier $(\mathbf{c}, \mathbf{p}, r)$, if $lockall(\mathbf{c}, \alpha) \wedge lockall(\mathbf{p}, \beta) \wedge r = k$, we say that $(\mathbf{c}, \mathbf{p}, r)$ is responsible for the key (α, β, γ), and thus hash items with this key is stored at this peer.

4.2 Routing in ComNET

The routing problem is to find a path to a peer with specified identifer or a peer that is responsible for a given hash key of a file. Similar to routing in Γ, routing in ComNET also proceed in 2 phases. Nevertheless, due to high dynamic of P2P network, fault tolerance should be introduced. Forwarding operations at a peer in ComNET are shown in algorithm 2 - 5.

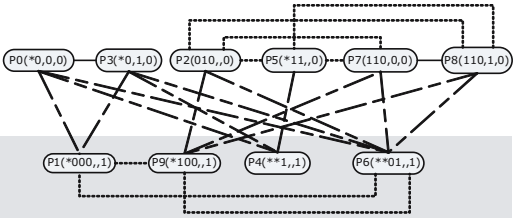


Fig. 1. Example of ComNET topology, where $k = 2, l_c = 4$

1. *Basic Routing Algorithm*: The routing algorithm corrects one bit of differences between the identifier of the source and the destination per step. The routing from P0 to P8 in figure 1 can be visualized as P0 → P3 → P1 → P9 → P8. Note that the resulting query path might not be the shortest one. But simple optimization can be obtained based on our routing algorithms. Since there is always more than one neighbor of the current peer whose identifier satisfies these algorithms. In this case, if we choose the neighbor with maximal *locking set* as next-hop, the length of query path can be reduced.

Input : The current peer $(\mathbf{c_1}, \mathbf{p_1}, r_1)$ and destination identifier (α, β, γ)
Output: The identifier $(\mathbf{c_2}, \mathbf{p_2}, r_2)$ of the next-hop peer along $Link_r$
1 r_2 is an integer that does not satisfies $\texttt{APlock}(\mathbf{c}_1, \alpha, r_2, k)$ or $\texttt{APlock}(\mathbf{p}_1, \beta, r_2, k)$. Non-$\gamma$ integers are preferred;
2 $\mathbf{c}_2$ satisfies $\texttt{APlockset}(\mathbf{c}_2, \alpha, k) \supseteq \texttt{APlockset}(\mathbf{c}_1, \alpha, k)$;
3 $\mathbf{p}_2$ satisfies $\texttt{APlockset}(\mathbf{p}_2, \beta, k) \supseteq \texttt{APlockset}(\mathbf{p}_1, \beta, k)$;

Algorithm 2. Finding a next-hop along $Link_r$ in ComNET: LinkRNextHop

Input : The current peer $(\mathbf{c_1}, \mathbf{p_1}, r_1)$ and destination identifier (α, β, γ)
Output: The identifier $(\mathbf{c_2}, \mathbf{p_2}, r_2)$ of the next-hop peer along $Link_p$
1 $r_2 = r_1$;
2 $\mathbf{c}_2$ satisfies $\texttt{APlockset}(\mathbf{c}_2, \alpha, k) \supseteq \texttt{APlockset}(\mathbf{c}_1, \alpha, k)$;
3 $\mathbf{p}_2$ satisfies $\texttt{APlockset}(\mathbf{p}_2, \beta, k) \supseteq \texttt{APlockset}(\mathbf{p}_1, \beta, k) \cup \{r_1\}$

Algorithm 3. Finding a next-hop along $Link_p$ in ComNET: LinkPNextHop

Input : The current peer $(\mathbf{c_1}, \mathbf{p_1}, r_1)$ and destination identifier (α, β, γ)
Output: The identifier $(\mathbf{c_2}, \mathbf{p_2}, r_2)$ of the next-hop peer along $Link_c$
1 $r_2 = r_1$;
2 $\mathbf{c}_2$ satisfies $\texttt{APlockset}(\mathbf{c}_2, \alpha, k) \supseteq \texttt{APlockset}(\mathbf{c}_1, \alpha, k) \cup \{r_1\}$;
3 $\mathbf{p}_2$ satisfies $\texttt{APlockset}(\mathbf{p}_2, \beta, k) \supseteq \texttt{APlockset}(\mathbf{p}_1, \beta, k)$

Algorithm 4. Finding a next-hop along $Link_c$ in ComNET: LinkCNextHop

```
   Input  : The current peer's identifier (c1, p1, r1) and destination identifier
            (α, β, γ)
   Output: The next-hop's identifier (c2, p2, r2)
 1 if (lockall(c1, α) ∧ lockall(p1, β) ∧ r1 = γ) then
 2     The destination has been reached
 3 else
 4     if (lockall(c1, α) ∧ lockall(p1, β)) then
 5         c2 satisfies APlockset(c2, α, k) ⊇ APlockset(c1, α, k) ;
 6         p2 satisfies APlockset(p2, β, k) ⊇ APlockset(p1, β, k);
 7         r2 := γ;
 8         if ((c2, p2, r2) is faulty) then
 9             The destination can not be reached temporarily
10         end
11     else
12         if (APlock(c1, α, r1, k) ∧ APlock(p1, β, r1, k)) then
13             (c2, p2, r2) := LinkRNextHop();
14             if ((c2, p2, r2) is faulty) then
15                 Find a non-faulty peer along Link_r that satisfies
                   |APlockset(c2, α, k)| + |APlockset(p2, β, k)| ≥
                   |APlockset(c1, α, k)| + |APlockset(p1, β, k)| −
                   RetreatThreadhold
16             end
17         else
18             if (¬APlock(p1, β, r1, k)) then
19                 (c2, p2, r2) := LinkPNextHop();
20                 if ((c2, p2, r2) is faulty ∧ ¬APlock(c1, α, r1, k)) then
21                     (c2, p2, r2) := LinkCNextHop();
22                     if ((c2, p2, r2) is faulty ∧ |APlockset(c1, α, k)| +
                       |APlockset(p1, β, k)| < (2k − 2)) then
23                         (c2, p2, r2) := LinkRNextHop()
24                     end
25                 end
26             else
27                 (c2, p2, r2) := LinkCNextHop();
28                 if ((c2, p2, r2) is faulty ∧ |APlockset(c1, α, k)| +
                   |APlockset(p1, β, k)| < (2k − 2)) then
29                     (c2, p2, r2) := LinkRNextHop()
30                 end
31             end
32         end
33     end
34 end
```

Algorithm 5. Routing in ComNET routeDHT

2. *Robustness*: In order to improve the robustness of our system, line 8-10, 14-16, 20-25 and 28-30 in algorithm 5 are added. The robustness-ensuring mechanism is triggered when the next-hop peer identified by the basic routing algorithm

is not working. The basic idea of our robustness-ensuring algorithm is to find a non-faulty next-hop peer with non-descending locking set, that is to say the size of the locking sets of the next-hop peer's group and intra-group identifiers should be greater or at least equal to those of the current peer respectively.

5 Performance Evaluation

In this section, some important performance metrics are measured by system simulation. All evaluation is performed within a single process with no network communication actually exists. System parameters $k = 3$ and $l_c = 8$ are used for all network size because they are fixed at programming time, and thus could not be adjusted according to the network size at runtime.

5.1 Query Path Length

Figure 2 plots the average and maximum query path length of ComNET as a function of the number of peers. There are two types of simulation: one is to find the number of hops required for routing between two randomly selected peers; the other is to find the length of routing path between two random selected peers with the same group identifier. We can see from figure 2 that when the number of peers reaches around 4k, the maximum length of routing path and intra-group routing path are fixed at 9 and 6 respectively. It can be explained as follows. According to proposition 2, the diameter of ComNET is related to k through the equation $Diameter = 3k = 9$. Also it is not difficult to verify that the maximum length of intra-group routing is 6. Note that, the parameter k is fixed at programming time, which means that the upper bound of routing path length in ComNET is a constant $\mathcal{O}(3k)$. Figure 3 plots the distribution of routing path length for a network size of 2^{22}. It can be seen from this figure that the length of 89.39% random routing varies from 6-9, and the length of 94.88%

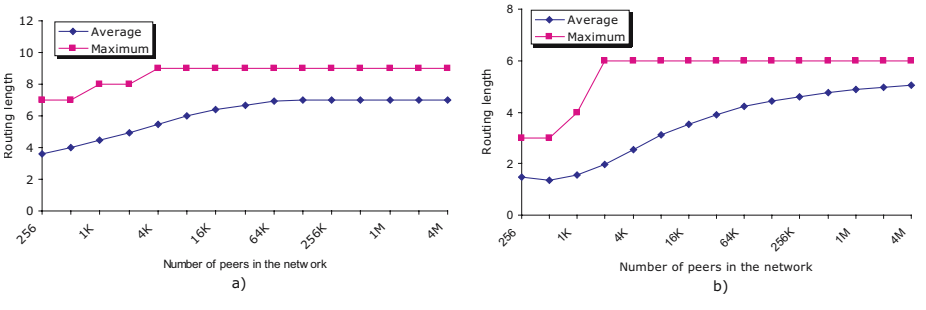


Fig. 2. Routing path length with different network size. a) plots the length of routing path between randomly selected peers and b) plots the length of routing path between randomly selected peers with the same group identifiers.

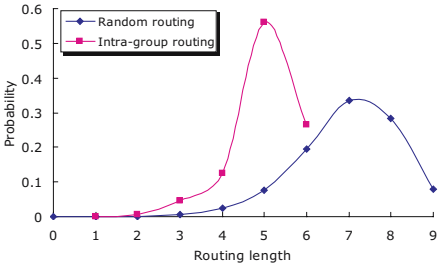


Fig. 3. The distribution of query length with 2^{22} peers

intra-group routing varies from 4-6. Low variance of routing path length may indicate low network jitter which is essential to real time P2P applications.

5.2 Size of Routing Table

Figure 4 plots the average size of routing table with different number of peers. According to proposition 1, the size of the routing table of a peer, is related to r_c and r_p. Thus the size of routing table would increase as the network size increase. When 2^{20} peers exist in ComNET, the average size of routing table is 21.3.

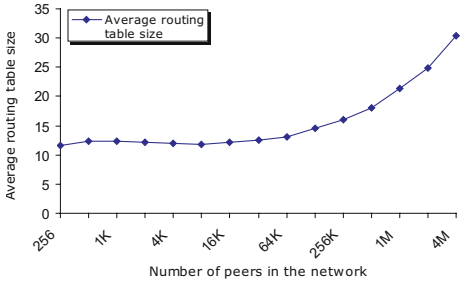


Fig. 4. The size of routing table

5.3 Robustness

In this section, we compute the probability that a query ends in failure and the average length of the successful routing in ComNET. All these tests are run in a network with 2^{22} peers. Figure 5a) plots the probability that routing ends in failure as a function of the percentage of peer failures. For the same failure rate, the probability that the ComNET routing algorithm exits in failure is lower than in Ulysses and Chord. Figure 5a) also plots the percentage of intra-group routing failure. As compared to random routing, the failure rate of intra-group routing is higher (57.2% when the peer failure rate is 20%), but is still low as compared

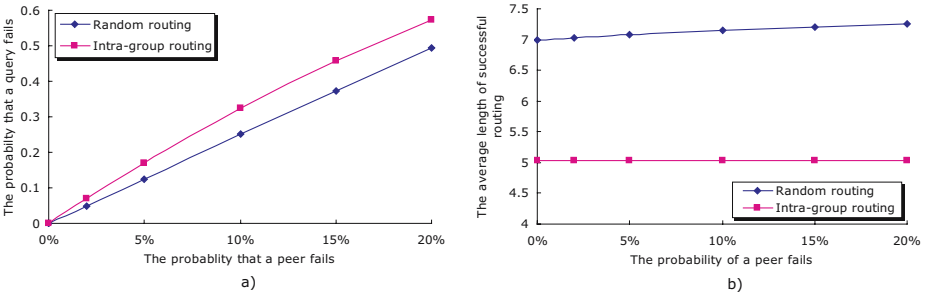


Fig. 5. a) the probability that routing ends in failure. b) the average length of successful routing. The network size is 2^{22}.

to that in Ulysses and in Chord. Figure 5b) plots the average hops required for the successful routing. The even curve in this figure indicates that the routing length is rarely influenced by faulty peers. As compared to ComNET, the routing length increases very remarkably in Ulysses (from 6.8 to 8.8, growing by 30%).

6 Conclusion and Future Work

In this paper, we define a new cayley graph Γ with small-world features. The essential properties including degree, diameter and CC1 show that Γ is a small-world graph and a suitable static model for P2P DHT network. In the latter sections, the protocols of P2P DHT network ComNET are proposed to capture the static structure in high dynamic environment. In ComNET, excellent routing performance is obtained while keeping the routing table small. Furthermore, the robustness of ComNET is also better than that of other protocols like Ulysses and Chord. And what is more, explicit peer grouping mechanism in ComNET enables us to implement effective resource browsing service in P2P DHT network.

Our further work will focus on how to adjust the number of regions (that is k) according to the network size, balancing zone splitting, finding a way to cluster a peer to more than one group and implementation of ComNET.

References

1. Kumar, A., Merugu, S., Xu, J., Yu, X.: Ulysses: A robust, low-diameter, low-latency peer-to-peer network. European transaction on telecommunications 15(6), 571–587 (2004)
2. Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A scalable and dynamic emulation of the butterfly. In: Proc. of ACM PODC 2002, ACM Press, New York (2002)
3. Xu, J.: On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. In: Proc. of IEEE Infocom 2003, vol. 1-3, pp. 2177–2187 (2003)
4. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature (1998)

5. Aberer, K., Alima, L.O., Ghodsi, A., Girdzijauskas, S., Haridi, S., Hauswirh, M.: The essence of p2p: a reference architecture for overlay networks. In: Fifth IEEE International Conference on Peer-to-Peer Computing (2005)
6. Sripanidkulchai, K., Maggs, B., Zhang, H.: Effcient content location using interest-based locality in peer-to-peer systems. In: Proc. of IEEE Infocom 2003: The conference on computer communications, vol. 1-3, pp. 2166–2176 (2003)
7. Xiao, W., Parhami, B.: Cayley graph as models of deterministic small-world networks. Information Processing Letters 97(3), 115–117 (2006)
8. Qu, C., Nejdl, W., Kriesell, M.: Cayley dhts - a group-theoretic framework for analyzing dhts based on cayley graphs. In: Cao, J., Yang, L.T., Guo, M., Lau, F. (eds.) ISPA 2004. LNCS, vol. 3358, Springer, Heidelberg (2004)

Data Grid Model Based on Structured P2P Overlay Network

Wei Song[1,2], Yuelong Zhao[1], Wenying Zeng[1], and Wenfeng Wang[1]

[1] School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China
[2] Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, China
color_unsw@126.com, ylzhao1@scut.edu.cn

Abstract. Data Grid provides integrated view of distributed data scattered across networks. Current Data Grid systems are centrally controlled. In this paper, we present a structured P2P based Data Grid model(P-DataGrid Model, PDG) which makes use of construction and routing algorithms of P-Grid ,a structured P2P system . PDG is organized as virtual multi-branch tree with binary tree as main body. Formal description of PDG is firstly introduced. Then we discuss the realization issues of PDG such as establishment of model, data storage service, information service, etc. Among these issues, our emphasis is on joining of nodes, registration and location of replica. Furthermore, we analyze the successful probability of location. Constructing Data Grid on structured P2P overlay can bring great advantages of scalability, decentralized control and reliability.

1 Introduction

Data Grid is a type of grid system which integrates data scattered across networks, providing virtual data storage. In currently deployed Grid systems, resources are often owned by research centers, public institutions, or large enterprises: in such organizations hosts and resources are generally stable [1]. In these systems, control of resources is centralized and usually handled by system administrators, which hinders dynamic and scalable expansion of the Grid infrastructure and resources [2]. Also, most Data Grid systems are centrally controlled and do not support dynamic data replica management, Such as Globus toolkit provides simple centered replica catalog service, and does not provide dynamic data replica creation, selection and consistence management [3]; SRB uses center server to provide replica location service [4].

As opposed to Grid, P2P systems integrate low-end resources which are also called "desktop at the edge of the Internet [5]" into unified super computing power and storage power. But nodes and resources are very dynamic. They frequently join and leave. So, current P2P technologies, especially resource discovery and location mechanisms, are more efficient in dynamic environment. Those technologies depend on P2P logical topology which can be classified into unstructured type and structured type. Unstructured P2P systems, such as Gnutella [6], resolve search requests by

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 282–291, 2007.

flooding techniques. Whereas Structured P2P systems use distributed hash tables (DHT) , for example Chord [7], Pastry[8], P-Grid, to establish a certain relationships among nodes and files. A DHT stores pairs (key, data) for distributed storing to enable fast searching of data when a key is given which results in better scalability, community efficiency and routing reliability.

We believe in the future the resource-sharing environment will include not only high-performance nodes, such as cluster, storage system, database, scientific instrument, but also low-end nodes, that is, Grid and P2P systems will converge in a unified resource-sharing environment [9]. This paper proposes a structured P2P based Data Grid model which is called P-DataGrid Model and PDG for short. It is based on P-Grid architecture, a P2P information system. Virtual binary tree constitutes main body of PDG with multi-branch in low layers. By introducing P2P discovery and location mechanism into Data Grid, data management, such as replica creation, selection, location and consistence management, will be more dynamic, scalable and less centralized.

The remainder of the paper is organized as follows. Section 2 introduces related work. Section 3 focuses on architecture of PDG and formal description is given. Section 4 discusses realization issues of PDG. Joining process, replica registration and location are elaborated. Section 5 analyzes the successful probability of replica location. The final section is the conclusion and future work.

2 Related Works

Our Data Grid model is mainly based on P-Grid system. P-Grid is a next generation peer-to-peer platform for distributed information management. Papers [10-12] about

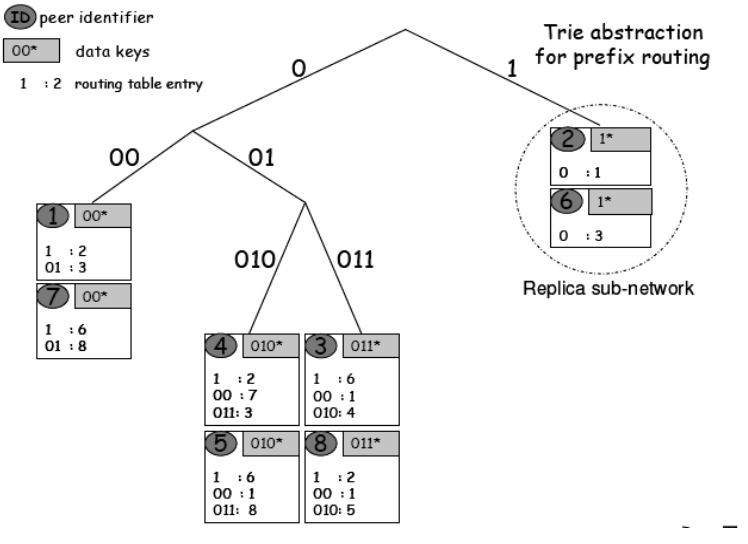


Fig. 1. Example P-Grid: Each peer is responsible for part of the overall tree [12]

P-Grid present the construction and routing algorithms on virtual binary search tree. By P-Grid construction algorithm in [10], peers construct the binary tree by pair-wise random interactions dividing gradually the key space in partitions defined by binary string the so-called peers' paths (denoted by *path* (peer)). Every peer takes over responsibility for one partition. Each peer records two kinds of information. One is that each peer (denoted by *a*) maintains a set of references to the other peers that store data items indexed by keywords *k* for which path (*a*) is a prefix. For example, in **Fig.1**, path of peer 4 and peer 5 is 010. Peer 4 and peer 5 will store references to all the data items with index of prefix 010. The other is a routing table. Each item refers to the address of at least one other peer that is responsible for the other side of the binary tree at that level. Thus, if a peer receives a binary query string it cannot satisfy, it must forward the query to a peer that is "closer" to the result. In **Fig.1**, *path* (4) =010, *path* (3) =011. We can see peer 3 is responsible for the other side of the binary tree at level three. So one item of peer 4 routing table is *011:3*, which mean if peer 4 receives request like 011*, it will forward the request to peer 3.

3 Conceptual Model of Data Grid Based on Structured P2P Overlay Network

3.1 PDG Overview

We introduce structured P2P overlay into Grid, and construct main body of PDG as a virtual binary tree. There are three types of node in PDG: Global Grid Node(GGN) , Local Grid Node(LGN)，Normal Grid Node(NGN).

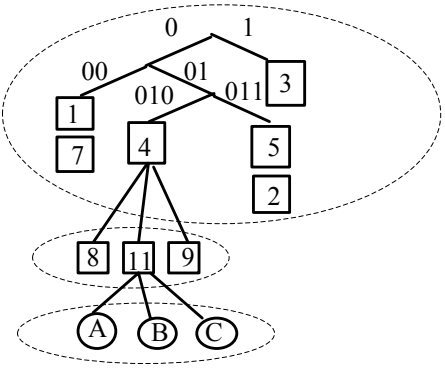


Fig. 2. Example PDG: Nodes are classified into three types

As showed in the **Fig. 2**, nodes in large circle are GGN which form the virtual binary tree by P-Grid construction algorithm. Nodes in small circle are LGN which are managed by a certain GGN. NGN is on the low layer. GGN and LGN manage the Grid, provide storage resources and also consume storage. NGN mainly consumes storage and sometimes provides storage if it is willing to do.

3.2 Formal Description of PDG

Definition 1. Grid node (G) is the node which is a member in PDG. There are three types of nodes: Global Grid Node(GGN) , Local Grid Node(LGN), Normal Grid Node(NGN).

Definition 2. GGN is reliable and high performance node which can be provided by research center, public institute and large company. GGNs work in controlled environment. Also normal users can offer their high performance resources voluntarily. GGN manages information of LGN and data file which have the same prefix with GGN's path. Only GGN has a path formed in the process of P-Grid construction. Each GGN maintains five tables.

(1) Global Routing Table. This table forms during the construction of virtual binary tree. Each item refers to at least one other node which is responsible for the other side of the binary tree at that level. We assume that each *GGN* is described by a multi-tuple of attributes (denoted by *ninf*) which contains node's name, address etc. For any *ninf* we define *node(ninf)=GGN iff getinf(GGN)=ninf*.

Table items also can be organized into a sequence $(l_1, Ninf_1)$ $(l_2, Ninf_2)$ … $(l_n, Ninf_n)$, where $l_i \in \{0, 1\}$ and $Ninf_i$ is a set of *ninf*. We define *path (GGN)* $=l_1 \ldots l_n$, *prefix (i, GGN)* $= l_1 \ldots l_i$ for $1 \le i \le n$ and *refs (i, GGN)* $= Ninf_i$. The sets $Ninf_i$, $1 \le i \le n$ are references to other nodes and satisfy the following property:

ninf $\in$ *refs(i, GGN)*: *prefix (i, node(ninf))= prefix(i-1, GGN)* $l_i^{\sim}$

where $l^{\sim}$ is defined as $l^{\sim} = (l+1) \bmod 2$.

(2) Local Routing Table. Each item is a 3-tuple (LGN, Key_{LGN}, $ninf_{LGN}$) which records the information of LGN. Key_{LGN} denotes hash value of LGN. $ninf_{LGN}$ denotes attributes of LGN including address.

Manage (a,b) denotes *a* manages *b*. *Equal(a,b)* means *a* is equal to *b*.

$\forall x \forall y(\exists i(Equal(path(y), prefix(i, Key_x))) \rightarrow Manage(y,x)), x \in \text{LGN}, y \in \text{GGN}$

(3) Replica Node Table. Replica node is redundant node of GGN. In **Fig.1**, node 4 and node 5 is redundant node each other.

Let *Replica (a,b)* denote *a* and *b* are redundant nodes.

$\forall x \forall y(Equal(path(y), path(x)) \rightarrow Replica(y,x)), x \in \text{GGN}, y \in \text{GGN}$

(4) LGN Data File Table. Each item is a 3-tuple ($Key_{DataFile}$, Key_{LGN}, $Attr_{DataFile}$), which records the relationship between data file and LGN where data file saved. $Key_{DataFile}$ denotes hash value of logical file name. $Attr_{DataFile}$ denotes attributes of file.

$\forall x \forall y \forall z (\exists i (Equal(path(y), prefix(i, Key_x)) \wedge Manage(z,x)) \rightarrow Manage(y,x)),$

$x \in \text{FILE}, y \in \text{GGN}, z \in \text{LGN}$

(5) GGN Data File Table. Each item is a reference ($Key_{DataFile}$, Key_{GGN}, $Attr_{DataFile}$)to the other node that stores suitable data file.

$\forall x \forall y \forall z (\exists i (Equal(path(y), prefix(i, Key_x)) \wedge Manage(z,x)) \rightarrow Manage(y,x)),$

$x \in \text{FILE}, y \in \text{GGN}, z \in \text{GGN}$

Definition 3. LGN is node with better performance. It finds a suitable GGN and joins the tree according to its hash value Key_{LGN}. LGNs managed by the same GGN can be replica nodes each other.

$\forall x \forall y(\exists i(Equal(path(y), prefix(i, Key_x))) \rightarrow Manage(y,x)), x \in \text{LGN}, y \in \text{GGN}$

$\forall x \forall y \forall z (Manage(y,x) \wedge Manage(z,x) \rightarrow Replica(y,z)), z \in \text{LGN}, y \in \text{LGN}, x \in \text{GGN}$

There are three main tables in each LGN.

(1)NGN Routing Table. It records information of NGNs it managed. Each item is a 2-tuple (NGN, $ninf_{NGN}$)

(2)NGN Data File Table. It records information of data file in NGN it managed. Each item is a 3-tuple ($DataFile$, NGN, $Attr_{DataFile}$).

(3)Local Data File Table. It records information of data stored in its own.

Definition 4. NGN is normal performance node. NGNs provide some storage ability, and mainly use the storage resource provided by Grid. Each NGN has a local data file table which contains its own data file information. In addition each NGN knows its management LGN after it joins in Grid.

Definition 5. Transition among each type of node. Performance of each node is announced by itself initially, and its type is changed when its performance parameters change. Each node has a performance monitor which can detect the load of it and change its type when some values exceed given threshold. Load balancing can be obtained in this way. Detailed transition mechanism will be designed in future work.

Definition 6. PGD is a multi-branch virtual tree. GGNs form the main body, binary virtual tree, which is established by P-Grid construction algorithm. LGNs join in the tree and choose suitable GGN nodes by the same algorithm. NGNs form multi-branches.

4 Realization of PDG

Realization of PGD includes two aspects. One is the establishment of model, such as how to initiate the tree, how to re-construct (join, leave, transit) the tree dynamically. The other is service based on the model, such as data storage service, information service, trust, quality of service, incentive mechanism. We will discuss those below.

4.1 Join and Leave

Flexibility of nodes' joining in and out shows dynamic and scalability of Grid model. There are two ways to join in Grid. One is entry point server which returns current available nodes to be entry nodes. Another is out-of-band method which is used when entry nodes are not available, such as email or communicating face to face. In PDG, we use the idea in P-Grid that nodes meet randomly, no matter why they meet, because they are involved in other operations, or because they systematically want to build the Grid [10]. In another words, all nodes can serve as entry point to the network, that is, entry point may be GGN, LGN or NGN. Considering different type of nodes will meet other different type of nodes, we design three functions: JoinGGN(entryGGN, newnode), JoinLGN(entryLGN, newnode), JoinNGN(entryNGN, newnode). Only pseudo code of JoinNGN is given in this paper to illustrate the detail process when entry node is NGN.

```
JoinNGN(entryNGN, newnode){
    Case newnode
GGN: get LGN(denoted by g1) which manages entryNGN;
       get GGN which manages g1;
       use P-Grid construction algorithm;
LGN: get LGN(denoted by g1) which manages entryNGN;
       get GGN(denoted by g2) which manages g1;
       find GGN(denoted by g3) which is prefix of Key_newnode from g2 using P-Grid
routing algorithm;
       write one item(newnode, Key_newnode, ninf_newnode)in local routing table of g3;
NGN: get LGN(denoted by g1) which manages entryNGN;
       write one item (newnode, Attr_newnode) in NGN routing table of g1;
if newnode has some data files to share
then register which will be explained in section 4.2 ;}
```

Fig. 3. Pseudo code of joining process when entry node is NGN

Leaving is the opposite process of joining. No matter what kind of node leaves, there are mainly two things to be considered. One is to delete or set unavailable of some items in related tables of its management nodes. The other is to transfer its own managed information of other nodes or data files to suitable nodes.

4.2 Data Storage Service

Data storage service manages logical name space and supports abstraction of storage system. It adapts to the change of Grid environment, that is, it allocates, reclaims and manages distributed storage dynamically. There are some issues to be discussed.

Logical Name Space. It is used to establish global and persistent identities which can span multiple storage systems. Logical name space service can set up and maintain the reflection between logical name and physical file name, which brings transparence of position. Most Grids support to organize data files into layer directory structure. Virtual multi-branch tree structure of PDG is beneficial to layer management of resource, and also supports layer structure of logical name space.

Attributes of Logical Name Space. Those attributes include storage position of replica, local file name, user defined property. When files are registered into logical name space, related attributes are created synchronously [13]. In PDG, all the tables that each type of node maintains can manage attributes well.

Selection of Data Abstraction Level. Data, information and knowledge are three kinds of data entity in Data Grid [13]. The abstraction level of PDG is data, which integrates data storage resource in Grid, and provides unified storage abstraction, which makes user not aware of heterogeneity in underlying resource.

Data Storage Service on Demand. Most Data Grids provide global data storage abstraction, which integrates all the storage resource in Grid into a whole storage space to user. But in practical application, it is not necessary. First, to normal users, they enter Grid randomly, and demand for storage resource is limited and temporary. For using resource efficiently, we should provide optional storage service for user so as to build user's local data storage view. Second, it is not always to use resource free.

Considering paid use, users can only get what they can pay for. In PGD, the character of multi-branch will satisfy different users' demands. Different local data storage view can form in a group of sub-tree. For users, they can customize their storage space, and this a way to allocation on demand.

Replica Management. To increase the usability and reliability of files, it is necessary to design good replica management, which includes replica creation, registration, allocation, location, selection and consistency. Here, we only analyze registration and location. The remainder problems will be discussed in the other paper.

Registration of Replica. Each node type can create replica itself when necessary. Information of replica must be registered in related management node. Replica created by different node type will have different register process. The whole process is described in Fig. 4.

common_prefix() means to get common prefix of two binary sequences.

Path() means to get a certain node path in virtual binary tree.

```
Register (DataFile, CreateNode){
 If (CreateNode is NGN) {
   find LGN which manages CreateNode ;
   write one item(DataFile,CreateNode,Attr_DataFile)in NGN data file table of found LGN;
   find GGN whose path equals common_prefix(Path(GGN), Key_DataFile);
   add one item (Key_DataFile, Key_LGN, Attr_DataFile) in LGN data file table of found GGN;
  }
 If (CreateNode is LGN){
   find GGN whose path equals common_prefix(Path(GGN), Key_DataFile);
   add one item(Key_DataFile,Key_CreateNode,Attr_DataFile)in LGN data file table of found GGN;}
 If (CreateNode is GGN){
   find GGN whose path equals common_prefix(Path(GGN), Key_DataFile);
   add one item(Key_DataFile,Key_CreateNode,Attr_DataFile)in GGN data file table of found GGN;}
}
```

Fig. 4. Pseudo code of replica registration

Location of Replica. Suppose the request for a certain replica is from NGN. If the NGN has, then return. If not, request will be forward to LGN. If LGN not has, request will be sent to GGN. If GGN not has, the hash value of the file will be generated, and request will be sent to proper GGN using P-Grid routing algorithm. The whole process is described in Fig. 5.

4.3 Other Issues

Information service is an important component in any Grid infrastructure. It provides resource detection, description and monitoring. In PDG, all the tables in each node will play an important role in information service. Each PDG node has its own monitor to trail its resource state. When node state differs from threshold, it will change its type and adjust its resource accordingly. Most nodes in PDG are low-end and less reliable. Trust mechanism will be established to judge the creditability. Meanwhile bad deeds from malicious nodes can be restrained. Quality of service is one of the crucial

```
locate(DataFile,node){
  search local data of node ;
  if found return find;
    else case node
GGN: find a GGN(denoted by g) whose path= common_prefix(Path(g),Key_DataFile);
       if g has then return find
       else{
      get Key_LGN from item (Key_DataFile,Key_LGN,Attr_DataFile) in LGN data file table of g;
      get LGN using Key_LGN by routing algorithm ;
      locate (DataFile, LGN);}
LGN: get NGN from item (DataFile,NGN,Attr_DataFile) in its NGN data file table;
       If get then locate (DataFile,NGN);
       Else{get GGN which manages LGN;
            locate (DataFile,GGN);}
NGN: get LGN which manages NGN;
       locate (DataFile,LGN);
}
```

Fig. 5. Pseudo code of replica location

problems in Grid, which promises the best resource allocation. Incentive mechanism will relate what node can get with what it has offered. The more it has offered the more public resource it will get. What's more, it is the support of allocation on demand.

5 Analysis of Location Performance

We analyze the probability of a successful location.

Table 1. Notation used in the analysis

p	the probability that a GGN is online
q	the probability that a LGN is online
s	the probability that a NGN is online
$GGNrepmax$	the max number of replica GGNs
$LGNrepmax$	the max number of replica LGNs
$NGNrepmax$	the max number of replica NGNs
k	the depth of binary search tree
d_{grid}	the total number of data objects that can be stored in the Grid
i_{node}	the number of references to data items each GGN stored
N	the number of GGN

k is given by inequality $k \geqslant log_2(d_{grid}/ i_{node})$ and N is given by inequality $N \geqslant GGNrepmax * (d_{grid}/ i_{node})$ in paper [10].

The successful probability to find a GGN that is responsible for a specific search key starting the search at an arbitrary GGN is $(1-(1-p)^{GGNrepmax})^k$ according to paper [10].Now we consider a simple case, that is, request is started at GGN, and successful response also comes from GGN. The successful probability of location is $(1-(1-p)^{GGNrepmax})^k$. Assume that $d_{grid}=10^7$ data objects exist, that a reference costs at most 10 Bytes of storage, and every GGN is willing to offer 10^6 Bytes of storage for indexing, which means $i_{node}=10^5$. Further more we assume p =0.7, for GGNs are high

performance and stable nodes. Table 2 shows the constraint of each parameter, to get successful probability of over 99%. We can see when the max number of replica GGNs *GGNrepmax* is 10 and the depth of binary search tree *k* is less than 18, successful probability can reach 99%.

Table 2. The numerical relations among each parameter to reach successful probability of 99%

GGNrepmax	*k*	*N*	notes
1			will never reach probability of 99%
4	<=1	10^2*4	
10	<=18	10^2*10	
20	<=287975293	10^2*20	

Next, let us analyze a more complex case. Query is started from NGN, and successful response returns from NGN. Each LGN and NGN must know its management nodes. If not, it will turn into a half-connected status which is a situation we must avoid by some mechanism. To analyze simply, we suppose each LGN and NGN always know its management nodes. The successful probability to find a LGN is $1-(1-q)^{LGNrepmax}$ and the successful probability to find a NGN after locating a proper LGN is $1-(1-s)^{NGNrepmax}$. The total successful probability is

$(1-(1-q)^{LGNrepmax})*(1-(1-p)^{GGNrepmax})*(1-(1-p)^{GGNrepmax}))^{k}*(1-(1-q)^{LGNrepmax})*1-(1-s)^{NGNrepmax}$

$= (1-(1-q)^{LGNrepmax})^{2}*(1-(1-p)^{GGNrepmax}))^{k+1}*(1-(1-s)^{NGNrepmax})$

Assume q=0.4, s=0.2, $GGNrepmax$=10, k=9, to get the successful probability of 99%, we can deduce that *LGNrepmax* must be more than 11 and *NGNrepmax* must be more than 27. Combining the two cases, we can see that it is very reasonable for PDG to have the size of one thousands GGNs, 11 replica for each LGN and 27 replica for each GGN.

6 Conclusions and Future Work

In this paper we introduce a new Data Grid model which takes advantage of structured P2P construction and routing mechanisms. Our model organizes resources in tree architecture which reflects the natural relationship in Grid and makes hierarchy management easily. Additionally, it greatly lessens the loads of center nodes and increases searching efficiency. Besides decentralization, also scalability and reliability are the great benefits.

Scalability: P-Grid construction algorithm promises scalability.

Reliability: From section 5 we have the idea that given a certain size of Grid, location algorithm always ensures high successful probability of node to response. Meanwhile, replica nodes have the same tables and data, which also improve reliability of system.

The work presented in this paper is a first step. Now, we are doing simulation experiments to analyze related theory and algorithm. Next step, we introduce security into PDG, including trust, reputation, authenticity, confidentiality and integrity. These improvements would make PDG a decent environment for collaboration among

multiple organizations, institutions and corporations. We also intend to address allocation on demand and on what nodes had contributed, which utilizes concept of market competition to maximum benefits that each node gets.

Acknowledgments

This work is supported by National Natural Science Foundation of China with the Grant NO.60573145, Natural Science Foundation of Hunan Province with the Grant NO.05JJ30120 and Science Foundation Program of Guangzhou with the Grant NO.2007J1-C0401.

References

1. Mastroianni, C., Talia, D., Verta, O.: A Super-peer Model for Resource Discovery Services in Large-scale Grids. Future Generation Computer System 21, 1235–1248 (2005)
2. Lamehamedi, H., Szymanski, B.K.: Decentralized Data Management Framework for Data Grids. Future Generation Computer Systems 23, 109–115 (2007)
3. The Globus Data Management Group.A Replica Management Service for High-Performance Data Grids, http://www.globus.org/datagrid/deliverables/ReplicaManagementService.pdf
4. Baru, C., Moore, R., Rajasekar, A., et al.: The SDSC Storage Resource Broker. In: CASCON 1998, Toronto, Canada (1998)
5. Shirky, C.: What is p2p... and What isn't? http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html
6. Clip2. The Gnutella Protocol Specification v0.4 (Document Revision 1.2) (June 2001), http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
7. Stoica, I., Morris, R., Karger, D., Frans, M., Kaashoek, Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: ACM SIGCOMM 2001 (2001)
8. Rowstron, A., Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. In: Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (2001)
9. Iamnitchi, A., Foster, I.: A Peer-to-Peer Approch to Resource Location in Grid Environments, http://www.csee.usf.edu/ anda/papers/iamnitchi-bookch.pdf
10. Aberer, K.: P-Grid: A Self-organizing Access Structure for P2P Information Systems. In: Proc. Int'l Conf. Cooperative Information Systems, Germany. LNCS, pp. 179–194. Springer, Heidelberg (2001)
11. Aberer, K., Punceva, M.: Improving Data Access in P2P Systems, http://www.p-grid.org/publications/papers/IC2002.pdf
12. Aberer, K., Datta, A., Hauswirth, M., Schmidt, R.: Indexing Data-oriented Overlay Networks, http://www.p-grid.org/publications/papers/VLDB2005.pdf
13. Berman, F., Fox, G., Hey, A.J.G.: Grid Computing: Making the Global Infrastructure a Reality. Wiley, New York (2003)

PeerTR: A Peer-to-Peer Terrain Roaming Architecture

Sheng Zheng[1], ZhanwuYu[2], Zhongmin Li[2], and Lu Gao[2]

[1] School of Electronic Information, Wuhan University, 129 Luoyu Road, Wuhan, Hubei, P.R. China, 430079
[2] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan, Hubei, P.R. China, 430079
{zhengsheng2006,yzw2008,zhongmli}@gmail.com, express_way@hotmail.com

Abstract. Since the large amount of terrain datasets, the terrain visualization component is separated from terrain datasets in spatial application systems in which terrain datasets are stored in the remote server and the terrain visualization component is running on the local host. For this kind of client/server architecture, the server can guarantee the certain response delay and fulfill the real-time rendering of large-scale terrain when the numbers of user are limited. With the numbers of clients quickly increase, the server performance will drop too rapidly to satisfy the client real-time rending need because the server loads are added linearly. This paper proposes a kind of architecture for terrain data transmission in peer-to-peer environment, called PeerTR, which realizes the large-scale terrain data transmission based on P2P by means of the broadband and storage resources of spatial application nodes. In PeerTR, the spatial application node, called TRPeer, is allocated certain size storage space for buffering terrain data received from server and other nodes. Meanwhile, buffers will reserve group member ID lists and the index list of terrain data. TRPeers in the same group share terrain data through exchanging the index lists. A prototype system is established and experimental results showed that PeerTR in performances such as the server's load and data transmit speed outperforms the mode of Client/Server unicast.

1 Introduction

Recently, many more high resolution terrain data referring to DEM and DOM can be acquired with the rapid development of spatial detect technique. The volume of this type of spatial application systems such as digital city and digital earth storing, processing terrain dataset is growing tremendously[1]. And the amount of terrain data is often up to GB even TB orders of magnitude. Since the large amount of terrain datasets, the terrain visualization component is separated from terrain datasets in spatial application systems in which terrain datasets are stored in the remote server and the terrain visualization component is running on the local host[2][3]. For this kind of client/server architecture, the server can guarantee the certain response delay and fulfill

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 292–300, 2007.

the real-time rendering of large-scale terrain when the numbers of user are limited. With the numbers of clients quickly increase, the server performance will drop too rapidly to satisfy the client real-time rending need because the server loads are added linearly.

IP multicast technology can be available to alleviate the load of the server by means of Multiplexing[4][5]. However, this kind of technology can't be applied widely because of some reasons such as realization complication, congestion control, reliability management and etc. P2P technology makes use of broadband, storage space and etc of network nodes to provide service for each other, which alleviate the server load and improve the efficiency of data transmission through appropriate arithmetic. At present, P2P is applied widely in IPTV[7][8], VOD[9], video meeting[10] [11]and etc.

This paper proposes a kind of architecture for terrain data transmission in peer-to-peer environment, called PeerTR, which realizes the large-scale terrain data transmission based on P2P by means of the broadband and storage resources of spatial application nodes. In PeerTR, the spatial application node, name TRPeer, is allocated certain size storage space for buffering terrain data received from server and other nodes. Meanwhile, buffers will reserve peer group member ID lists and the index list of terrain data. TRPeers in the same group share terrain data through exchanging the index lists.

In section 2, we describe the architecture of PeerTR and the interior architecture of TRPeer. In section 3, we give out the terrain data organization, the management and the fault-tolerant mechanism of group members, the operation of buffer mapand the scheduling algorithm of terrain data. In section 4, we show a terrain walkthrough prototype system called GlobeSIGht as the test bed, and give out the test results and analysis based on the comparison of server loads between the PeerTR system and C/S unicast system. In section 5, we give some conclusions.

2 Architecture

2.1 Architecture of PeerTR

In PeerTR, the P2P mode isn't used to replace the C/S mode but as complement. Terrain roaming node called TRPeer can acquire terrain data by means of two ways:fetching data directly from server;fetching data from other nodes.

Every TRPeer maintains a partnerlist and acquire terrain data from nodes in partnerlist.Server node is the partner of all TRPeer. TRPeer exchanges buffer metadata, which describes the information of buffered terrain data, with partners and then gain terrain data from parterns according to buffer metadata and scheduling algorithm.The figure 1 give the architecture of PeerTR in which broken lines represent the transmission of control message and real lines represent the transmission of terrrain data.In the figure 1,TRPeer 1_2, TRPeer 1_3, TRPeer 1_k and server is the partners of TRPeer 1_1.

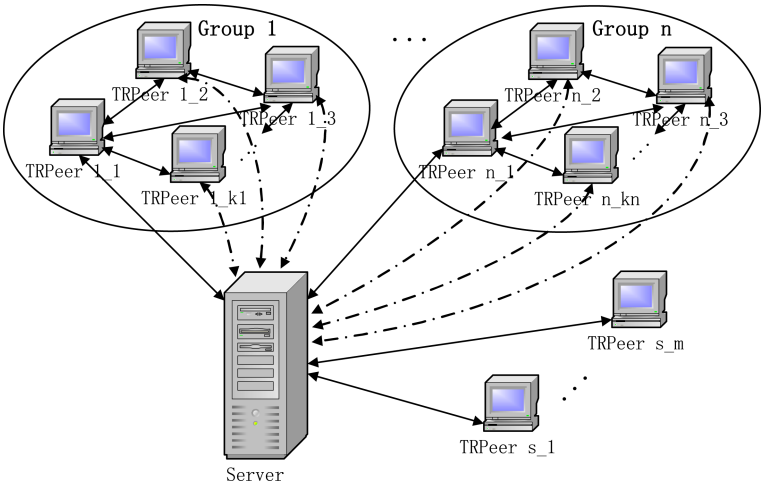


Fig. 1. Architecture of PeerTR

TRPeer 1_1 obtains terrain data from these nodes.The partnerlist isn't invarible and the TRPeer will continually adjust the partnerlist.

2.2 Interior Architecture of TRPeer

TRPeer (as showed in Fig. 2) consists of three modules mainly: Terrain Visualization Module(TVM), Terrain Data Receiving Module(TDRM), Terrain Data Sending Module(TDSM).

TVM schedules DEM (Digital Elevation Model) data block and DOM(Digital Orthophoto Map) data block from TBDB(Terrain block Data Buffer) and realize to model and render three-dimensional terrain. TDRM take charge of receiving terrain data from partners and major components in it involve the following functions:

- membership management 、 buffer metadata exchange and failure detecting;
- terrain data scheduling in which a provider for wanted data blocks will be elected from partnership in terms of buffer metadata of the partner;
- terrain data receiving in which the terrain data blocks requested will be parrallelly received from mutiple partners elected in above scheduling.

TDSM is responsible for sharing terrain data blocks in buffer with other partners and main components in it involve the following funcitons:

- receiving and processing requests from other TRPeers;
- transimitting the terrain data;
- sending message about the nodes leaving the system.

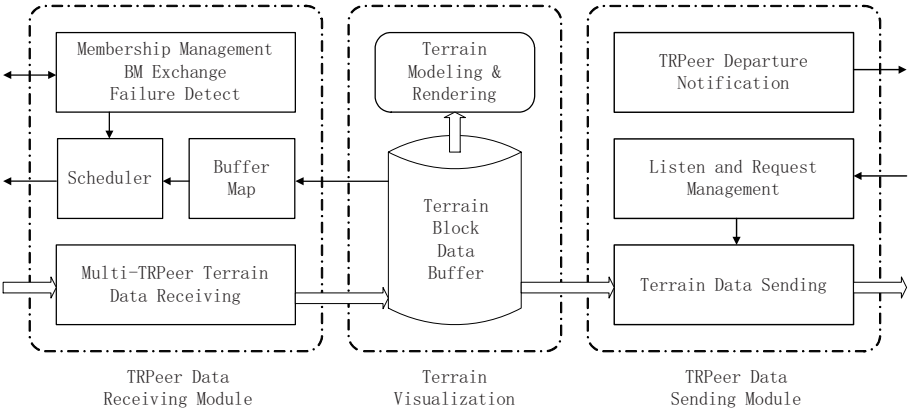


Fig. 2. Interior architecture of TRPeer

3 Implementations

3.1 Organization of Terrain Data

In the terrain visualization system,data organization based on pyramid model is widely used. Our system adopt the following method to organize and codec terrain data block.

1. Multi-resolution pyramid model based on multiple of 2;
2. Dividing the terrain data of every level into data block with equal interval and size.The data block divided is the basic unit to be transmit;
3. Data blocks are recorded in the form of *nLevel_nBlkNoX_ nBlkNoY* . *nLevel* records the level to which the block belongs in pyramid; *nBlkNoX* records line number and *nBlkNoY* records row number.

3.2 Management of Partner List

In the PeerTR, each TRPeer has a unique identifier and an IDCache caching other TRPeer ID. When a TRPeer enters into the PeerTR, it firstly sends a request to the server node and gets a response with a living TRPeer ID which is choosed according to optimal policy such as whether nearer the TRPeer requesting. Then the new entering TRPeer communicates with the TRPeer whose id is acquired from server and gains the partnerlist from it. Two data structure are defined in order to manage and maintain the IDCache as follows.

- DS1=< seq_num, id, count_partner，time_to_live >;
- DS2=< seq_num, id, count_partner，time_to_live ,last_update_time>.

TRPeer in system will leave or rejoin dynamically and the states of TRPeer will change at any moment. TRPeer broadcast a message containing its state to the partners periodly based on SCAM protocol[12].The message format is a four-tuple structure just like DS1.After it receives the message, TRPeer firstly lookup the record in its IDCache whose structure is just like DS2 to match the seq_num in the received message. If the seq_num is matched, the TRPeer will discard the message. Otherwise, the TRPeer will compare their id further. The record in the IDCache is updated if the ID exists in the IDCache and a new record is created in the IDCache while the ID doesn't exist.

The different sets of nodes satisfy different QoS requirements. It is the most important to keep the delay as low as possible in the real-time terrain rendering system. TRPeer should choose neighboring nodes physically to set up the transmission channel to satisfy the requirement. The nodes in the internet will not be selected when there are nodes providing terrain data in the intranet. Based on this principle, we designed an optimal policy to the group members. Its main thought is that TRPeer establishes a evaluating mechanism for partner in its partner list according to $\bar{s}_{i,j}$ periodly. $\bar{s}_{i,j}$ represents the average number of terrain block that node i obtained from node j in unit time. $\bar{s}_{i,j}$ can be calculate for the log in TRPeer and the bigger value of $\bar{s}_{i,j}$ show node i can get data more quickly. TRPeer will delete the smallest $\bar{s}_{i,j}$ in its partner list during each scheduling and fetch a new node ID from server.

Some fault-tolerant policies need be designed to guarantee the data transmitting to continuous because of the node will leave randomly. There is a server node in the partner list of TRPeer ,which is a backup node to all the running TRPeer nodes. Once the one of partners providing data fail to transmit data, the TRPeer quickly put forward requests to the server and the server will take over the failured nodes.

3.3 Operation of Buffer Map

Buffer map(BM) is used to record and index the terrain block in TRPeer buffer .Buffer map may be called as metadata about the buffer.Buffer maps are exchanged with each other among partners.TRPeer get to know the terrain data cached in the other partners through the buffer map. The terrain data is organized in the form of tile-pyramid model and data blocks among different levels is denoted in quadtree structure.

We design a kind of algorithm of buffered data mapping.Multiple quadtree will be set up in terms of the block number in the level 0. Each quadtree will correspond with a BM and each node in quadtree will be denoted with a bit 1 or 0. 1 denote this block data exists and 0 denote it doesn't exist.A byte is used to denote the row number and line number of node in quadtree. The algorithm for searching data block in BM is as following:

Setp 1. the required terrain number, nLayer_nBlkNoX_ nBlkNoY, is computed by terrain visualization module;

Step 2. computing the line X_0 and row Y_0 of the root of the quad tree;

$$X_0 = \frac{nBlkNoX}{2^{nLayer}} \tag{1}$$

$$Y_0 = \frac{nBlkNoY}{2^{nLayer}} \tag{2}$$

Setp 3. computing the decimal Morton code M_Q based on the step2;

Step 4. getting the result from $I_b = 8+4^{nLayer-1}+ M_Q$; the result is the bit of the BM that the required data is mapped in.

$$I_b = 8+4^{nLayer-1}+ M_Q \tag{3}$$

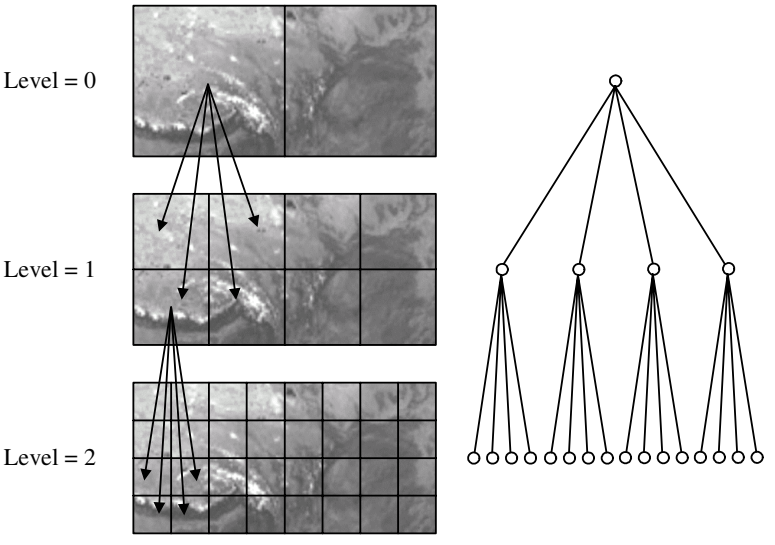


Fig. 3. Quad-tree model of terrain data

3.4 Scheduling of Terrain Block

The purpose to scheduling is to solve how to choose optimal partners to realize efficient and reliable data transmission in the PeerTR. The scheduling algorithm is very important to the system performance. The round-robin algorithm can satisfy the system requirement in the static and homogenous system. However the performance and state of TRPeer are different, so an algorithm is necessary to adapt with the dynamic enviroment. The algorithm must meet two requirements: the deadline and the response time.The requested data block must be obtained before the deanline and the response time should be as small as possible.

We design an intelligent scheduling algorithm to meet the two requirements above. The core thoughts in our algorithm is to calculate the number of nodes providing data download according to BM and download firstly the data blocks whose potential providers is less. Among the multiple potential suppliers, the one with the highest bandwidth and enough available time is selected.

4 Results

4.1 Test Bed

We have designed a test bed called GlobeSight, a large-scale terrain roaming system., TRPeers can upload or download data to from each other in the test bed. The system uses eight-level DEM data whose resolution is ninety-meter and terrain image data whose resolution is thirty-meter about China land area. The data is partitioned by 150*150 and the lowest-resolution image data is partitioned to four pieces. Each TRPeer has at most four BM lists. The network structure is illustrated in Fig. 4, there are three groups: group A and B are connected with the terrain data server by CERNET, and group C connected with terrain server by ADSL.

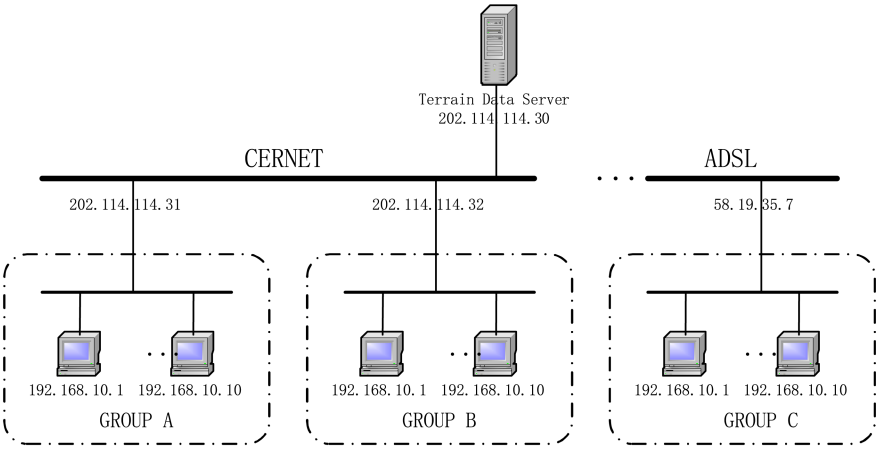


Fig. 4. Architecture of Test bed

4.2 Results and Analysis

In order to focus on the difference of the server load and data transmission efficiency between the PeerTR system and C/S unicast system, we make several assumptions to simplify the test configuration: firstly, the number of nodes in the experiment is fixed; secondly, the roaming line is uniform in every TRPeer, namely China—Hubei

Province—Wuhan City; the lastly, each TRPeer joined the system is random based on Poisson distribution. In the same configuration, we repeatedly experiments for 10 times and compute the average value of ten group of test results. Fig. 5 shows the cumulative distribution of the two systems for server loads respectively.n is the number of TRPeers and m is the number of working threads which measures the server load.

From the test results, we discover that PeerTR is more propitious to relieve the server load than C/S system. In the PeerTR system, when the number of nodes reaches 20, the server load reaches maximum, then keeps in a stable range, but not increasing linearly with the increasing of the number of nodes. According to Poisson distribution law, the probability is more than 96 percent when a new TRPeer joins the system after the number of nodes reaches 23, and in the three groups (group A, B and C), there are at least three TRPeers to join every group. The cache in the three TRPeers can generally accommodate a majority of data needed by the users when they roam. The users in the local network can download the data from the interior nodes, so the server load can keep in a steady range after the number of nodes exceeds 20.

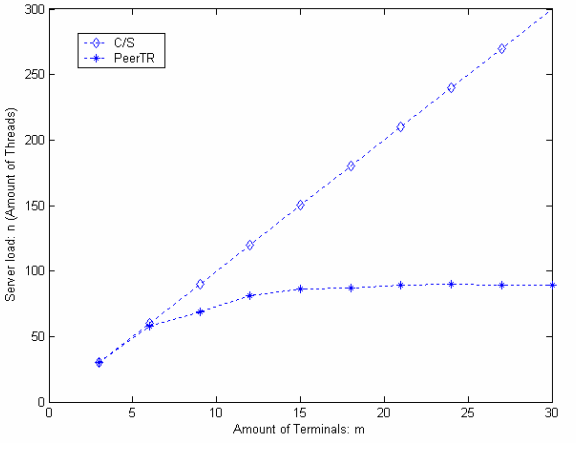


Fig. 5. Comparison of server loads between the PeerTR system and C/S unicast system

5 Conclusions

Massive data transmission is the key to realize terrain visualization based on the network. In this paper, we present PeerTR, a new architecture for terrain data transmission in peer-to-peer environment, which utilizes the clients to decentralize the terrain data distribution servers and to alleviate the server loads and network bandwidth loads. Furthermore, the terrain data can be downloaded from more than a node so that the download speed is greatly accelerated. Test results show that the mechanism can realize the massive data transmission at the lower cost compared with the single broadcast. So the PeerTR can be seen as a general architecture for terrain roaming system in the peer-to-peer network.

Acknowledgement

This work was supported by the National Key Basic Research and Development Program of China (No.2004CB318206).

References

1. Zhang, L., Zhang, Y., Yang, C.: Effective solutions to a global 3D visual system in networking environments. Science in China,Ser.D 48(11), 2032–2039 (2005)
2. Wang, G.-p.: The Implementation of High-bandwidth Network Based Real-time Interactive Browsing for 3D Massive Terrain Data [J]. Acta Geodaetica et Cartographica Sinica 31(1), 34–38 (2002)
3. Tu, Z., Liu, Y., Su, K.: A Modeling Method for Large Scale Terrain Oriented 3D Display[J]. Acta Geodaetica et Cartographica Sinica 31(1), 72–76 (2004)
4. Stephen, E.D., David, C.: Multicast Routing Datagram Internet works and Extended LANS. ACM Transactions on Computer Systems 8(2), 85–110 (1990)
5. Deering, S.E., Estrin, D., Farinacci, D.: The PIM architecture for wide-area multicast routing. IEEE Trans. Networking 4(2), 153–162 (1996)
6. Chu, Y.-H., Rao, S.G., Zhang, H.: A case for end system multicast. In: SIGMETRICS 2000, Santa Clara, CA, USA, vol. 6, pp. 1–12 (2000)
7. Tran, D., Hua, K.: An efficient peer-to-peer scheme for media streaming. In: Tran, D., Hua, K. (eds.) Proc.of the IEEE INFOCOM 2003, pp. 1283–1293. IEEE Computer Society Press, Los Alamitos (2003)
8. Xu, D., Hefeeda, M., Hambrusch, S., Bhargava, B.: On peer-to-peer media streaming. In: Proc. ICDCS 2002 (July 2002)
9. Guo, Y., Suh, K., Kurose, J., Towsley, D.: P2Cast:P2P patching scheme for Vod service. In: Proc.of the WWW 2003, pp. 301–309. ACM Press, New York (2003)
10. Zhang, X., Liu, J., Li, B., Yum, T.S.P.: DONet/CoolStreaming:A Data-Driven Overlay Network for Live Media Streaming. In: IEEE INFOCOM 2005. Miami,USA (2005)
11. Haigang, G., Ming, L., Yingchi, M.: Research Advances in Key Technology of P2P-Based Media Streaming[J]. Journal of Computer Research and Development 31(1), 2033–2040 (2006)
12. Ganesh, A.J., Kermarrec, A.M., Massoulie, L.: Peer-to-Peer Membership Management for Gossip-based Protocols. IEEE Transactions on Computers 52(2), 139–149 (2003)

SDRD: A Novel Approach to Resource Discovery in Grid Environments

Yiduo Mei, Xiaoshe Dong, Weiguo Wu, Shangyuan Guan, and Junyang Li

School of Electronic and Information Engineering, Xi′an Jiaotong University,
Xi′an, 710049, China
meiyiduo@gmail.com,
xsdong@mail.xjtu.edu.cn

Abstract. Scalability and routing efficiency are two crucial aspects of resource discovery in grid environments. SDRD, a novel approach to resource discovery in grid environments, is proposed in this paper. Basic model of resource discovery is proposed. Based on the basic model, formal description of SDRD is given. Sophisticated routing strategies are implemented in SDRD to improve the scalabilities and routing efficiencies of existing peer-to-peer based discovery approaches to grid resource. Improved Distributed Hash Tables (DHTs) are adopted within virtual organizations (VOs) to achieve better local routing efficiency. SuperNodes are only in charge of forwarding requests to other SuperNodes in different VOs when the desired resources are not present in local VO, which enhances the scalability due to the lower load on SuperNodes. SDRD could be applied to other resource management models to offer excellent scalability and high routing efficiency. Results of analysis and simulations show that SDRD scales well and could offer an efficient decentralized discovery service in grid.

1 Introduction

In essence, the grid resources are heterogeneous, dynamic and distributed [1]. These characteristics create significant difficulties for traditional centralized resource discovery services [2].

Peer-to-peer (P2P) systems are distributed systems consisting of interconnected nodes [3]. Reference [4] suggested that designers could use the P2P philosophy and techniques to implement decentralized grid systems, and the monitoring and discovery service (MDS) of Globus Toolkit [6] could be effectively redesigned using a P2P approach.

Scalabilities and routing efficiencies of existing P2P-based approaches to resource discovery would suffer from the drawbacks listed below: (1) some works employ unstructured P2P routing algorithms, thus the routing efficiencies are low and no locating guarantees are provided; (2) organization strategies of grid nodes in some works have not considered the physical topology; (3) some works use super nodes to provide the discovery service, but the load on super nodes in these works is high, thus super nodes could be potential bottlenecks. As the deployed grid size increases from

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 301–312, 2007.

tens to thousands of nodes, resources and requests will increase dramatically, because of the above problems, previous works would not scale well and their routing efficiencies are low.

To address the above problems, we propose SDRD, a **S**upernode and **D**HT (Distributed Hash Tables) based approach to **R**esource **D**iscovery in grid environments. Basic model of resource discovery is proposed. Based on the basic model, formal description of SDRD is given. Sophisticated routing strategies are implemented in SDRD to improve the scalabilities and routing efficiencies of existing P2P-based discovery approaches to grid resource. Improved DHTs are adopted within virtual organizations (VOs) to achieve better local routing efficiency. SuperNodes (super node of SDRD is represented by the term ″SuperNode″ in this paper) are only in charge of forwarding requests to other SuperNodes in different VOs when the desired resources are not present in local VO, which enhances the scalability due to lower load on SuperNodes. SDRD could be applied to other resource management models to offer excellent scalability and high routing efficiency. To demonstrate the applicability and efficiency of this approach, we presented experimental results got by using comparison methodology. Results of simulations show that SDRD scales well, and could offer an efficient decentralized discovery service in grid.

The rest of this paper is organized as follows. Section 2 summarizes previous works on P2P-based approaches to resource discovery in grid environments. Section 3 presents formal description of SDRD. In section 4, we introduce key technologies designed by SDRD. Model analysis is presented in section 5, and this section will show how to apply SDRD to other resource management model. Finally, we end this paper with discussions on the experimental results in section 6 and conclusion in section 7.

2 Related Works

Resource discovery is a hot topic in grid computing fields. Traditional approaches maintain hierarchically organized servers to index grid resource information. However, the centralized servers might become a bottleneck. Meanwhile, centralized approaches have the inherent drawbacks of a single point of failure [10].

Peer-to-peer based approaches to resource discovery in grid environments were proposed to address the above problems. Reference [2] proposed resource discovery mechanism based on unstructured P2P networks. However, this approach does not scale well due to the large amount of query messages generated by flooding. And unstructured P2P systems can not guarantee that resources could always be located. Reference [5] proposed a P2P-based data discovery mechanism similar to [2]. Reference [7] proposed a super-peer model for building resource discovery services, where a super-peer manages metadata associated to the resources provided by the nodes of a single VO. This approach might cause high load to the super-peers, which might become the bottlenecks and query hotspots.

In P2P fields, it is widely accepted that DHTs are the building blocks for next-generation large scale decentralized systems. Current implementations of DHTs are efficient for 1-dimensional queries. But, extending DHTs to support d-dimensional range queries in grids is a complex problem [8]. References [9], [10], [11], [12], [13],

[14] had proposed different solutions to this problem. These works ([9]-[14]) are referred to as improved DHTs in this paper, which means that these algorithms are based on DHTs and provide *d*-dimensional range query support. However, nodes forward queries according to well-defined rules without any knowledge of underlying physical topology might cause mismatch between P2P overlay network and the physical underlying network, which greatly limits the performance gain from various search or routing techniques [18]. SDRD alleviate the mismatch problem by organizing nodes according to their geographical proximity.

3 Formal Description of SDRD

In this section, we propose a basic model of resource discovery in grids at first. Then formal description of SDRD is depicted based on the proposed basic model.

3.1 Basic Model of Resource Discovery in Grids

The proposed basic model of resource discovery is inspired by the Boolean Model in the information retrieval field [19].

A basic model of resource discovery in grids is a sextuple /PS, QS, *F*, ZS, *Sf*, *T*/, where

Definition 1. *T* is a certain time interval, $t_{start} \le T \le t_{end}$, where t_{start} and t_{end} represent, respectively, the moment that the grid system is established and destroyed.

The grid is dynamic, which means available resources, available grid nodes, available services and user requests are variant at different time. To facilitate our description, Assumption 1 was given.

Assumption 1. The grid is relatively static during a short time interval T_0, where $t_{start} \le t_1 \le T_0 \le t_2 \le t_{end}$, see Fig. 1.

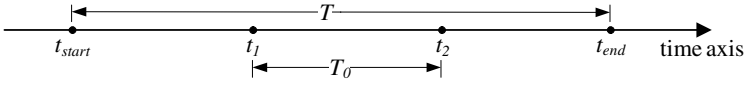


Fig. 1. Example of a short time interval T_0

Based on Assumption 1, the basic model is revised to the sextuple /PS, QS, *F*, ZS, *Sf*, T_0/.

Definition 2. PS is a set composed of logical views (or representations) for available resource providers in grid during time interval T_0.

Definition 3. QS is a set composed of logical views (or representations) for the user information needs in grid during time interval T_0. Such representations are called requests.

Definition 4. *F* is a mechanism for modeling resource providers, requests, and their relationships.

Definition 5. ZS is a set composed of resource providers which could satisfy request constraints on the node attribute values. ZS $\subseteq$ PS.

Expression F (PS, q) $\rightarrow$ ZS, where $q \in$ QS, represents that: through mapping process F, a resource request q could find a set of nodes ZS, which can provide service satisfying the request q within a given set PS.

Definition 6. *Sf* is a ranking function. Expression Sf : ZS $\rightarrow R^+$, where R^+ is the set of positive integers, represents associating a positive integer with a matched resource provider according to certain criterion. Such ranking defines an ordering among the providers with regard to the request.

3.2 Match Value of SDRD

Match value is a criterion which is used to check if a node could satisfy a request. In SDRD, each node belongs to one single VO. Suppose that there are totally n ($n \in R^+$) VOs in the grid. Let PS_i represent VO_i, then $\forall p$ (($p \in$ PS) $\Rightarrow \exists k$ ($(1 \le k \le n) \wedge (p \in PS_k) \wedge (PS_k \subseteq PS)$)). Furthermore, $PS_i \cap PS_j = \varnothing$, where $1 \le i, j \le n$ and $i \ne j$. Let $\prod = \{PS_1, PS_2, \ldots, PS_n\}$, $\prod$ is a partition of PS.

A request q ($q \in$ QS) is composed of different constraints on the node attribute values linked by three connectives: *not* ($\neg$), *and* ($\wedge$), *or* ($\vee$). A request composed of d kinds of constraints on the node attribute values is called d-dimensional request [8], where each constraint condition is represented by c_i ($1 \le i \le k$). Iff a resource provider p ($p \in$ PS) satisfies all the constraint conditions of a request q ($q \in$ QS), we can say that p can satisfy request q, otherwise, p can not satisfy q.

A request q is essentially a conventional expression which can be represented as a disjunction of conjunctive vectors (represented by q_{dc}), i.e. in disjunctive normal form – DNF [19], represented by q_{dnf} ($q_{dc} \in q_{dnf}$). The expression $p \Uparrow q_{dc}$ (q_{dnf}) represents p could satisfy q_{dc} (q_{dnf}), other wise, represented by $p \Downarrow q_{dc}$ (q_{dnf}).

Theorem 1. $\exists q_{dc}((q_{dc} \in q_{dnf}) \wedge (p \in PS) \wedge (p \Uparrow q_{dc})) \Rightarrow (p \Uparrow q)$ is true, i.e. if p could satisfy $q_{dc}(q_{dc} \in q_{dnf})$, then p could satisfy request q.

Proof. Without loss of generality, suppose that q (q_{dnf}) is represented by B = $\{q_{dc1}, q_{dc2}, \ldots, q_{dcn}\}$, define F: PS $\times$ QS $\rightarrow$ I, I = {*false*, *true*}, F represents if provider p ($p \in$ PS) could satisfy request q ($q \in$ QS). And let set A = {p}, then A $\subseteq$ PS, the Cartesian product is A $\times$ B = $\{(p, q_{dc1}), (p, q_{dc2}), \ldots, (p, q_{dcn})\}$, we define a function f : A $\times$ B $\rightarrow$ I, and f_i represents the degree that p could satisfy q_{dci}, i.e. if p could satisfy q_{dci}, then f_i = *true*, else f_i = *false*. For q, since $q_{dnf} = q_{dc1} \vee q_{dc2} \vee \ldots \vee q_{dcn}$, $F(p, q) = f_1 \vee f_2 \vee \ldots \vee f_n$. Because that $\exists q_{dc}((q_{dc} \in q_{dnf}) \wedge (p \in PS) \wedge (p \Uparrow q_{dc}))$, suppose that $p \Uparrow q_{dcm}$, where $q_{dcm} \in q_{dnf}$ and $1 \le m \le n$, so f_m = *true*. Therefore, $F(p, q)$ = *unknown* $\vee \ldots \vee$ *true* $\vee \ldots \vee$ *unknown* = *true*, where *unknown* $\in$ I. We could conclude that p can satisfy request q.

Base on Theorem 1, Formula 1 is given to calculate the match value between a request q and a resource provider p.

$$MatchValue_{(p,q)} = \begin{cases} 1\text{: if } \exists q_{dc}((q_{dc} \in q_{dnf}) \wedge (p \Uparrow q_{dc})); \\ 0\text{: other wise .} \end{cases} \quad (1)$$

3.3 Match Mechanism *F* of SDRD

Match mechanism *F* is the core part of SDRD. *F* is described by a rule-set LS_{rn}, which is composed of six rules with regard to regular nodes, see Fig. 2 (a), there are restrictions among the rules, for example, the results of Rule 1 might activate Rule 2 or Rule 3. This restriction is referred to as rule transition, denoted by arrow lines in Fig. 2. The rule transition process would terminate at the end of Rule 2, Rule 5 and Rule 6, and these three rules are depicted with black frame to indicate that they contain the terminal conditions of rule transition. The rule-set LS_{sn} to describe the SuperNodes′ match mechanism is shown in Fig. 2 (b).

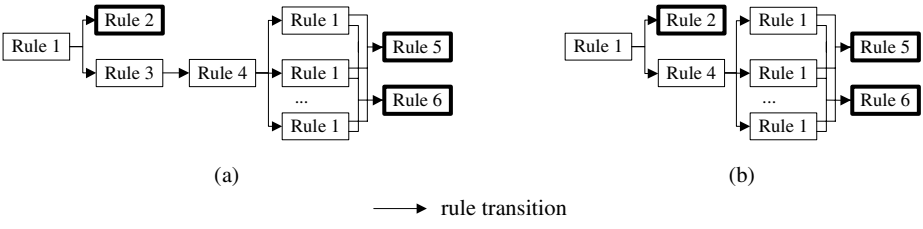


Fig. 2. Rule-set

A rule tells the system how to react to a particular situation. Each rule consists of two parts: condition and action, denoted by $X \Rightarrow Y$, where *X* is the condition and *Y* is the action. Explanations to the rules are as follows.

Rule 1. $\forall q(((q \in QS_{\text{local}}) \wedge (QS_{\text{local}} \subseteq \text{QS})) \Rightarrow \exists p((p \in PS_{\text{local}}) \wedge (PS_{\text{local}} \subseteq \text{PS}) \wedge (ImDHTs(Alg, q, p, ZS_{\text{local}})) \wedge (ZS_{\text{local}} \subseteq PS_{\text{local}})))$.

Rule 1 could be set as the beginning of rule transition with regard to LS_{rn}. If a request *q* is generated in or issued to local VO, node *p* will forward *q* according to well-defined rules, i.e. one of the improved DHTs, denoted by *Alg*. Node *p* is a grid node in local VO without considering if it is a SuperNode or not. The results will be saved in ZS_{local} by *p*.

Rule 2. $\exists p((p \in PS_{\text{local}}) \wedge (PS_{\text{local}} \subseteq \text{PS}) \wedge (MatchValue_{(p,q)} = 1)) \Rightarrow \exists ZS_{\text{local}}\ ((Return(ZS_{\text{local}})) \wedge (ZS_{\text{local}} \subseteq PS_{\text{local}}) \wedge Exit)$.

Rule 2 could express the following meanings: if the desired resources are located in local VO, then the discovery process will terminate and ZS_{local} are returned. $MatchValue_{(p,q)}$ is calculated using Formula 1.

Rule 3. $\forall p(((p \in PS_{\text{local}}) \wedge (PS_{\text{local}} \subseteq \text{PS}) \wedge (MatchValue_{(p,q)} = 0)) \Rightarrow \exists SN_m((SN_m \in PS_{\text{local}}) \wedge (Forward(q, SN_m)) \wedge (ZS_{\text{local}} = \varnothing)))$.

If the desired resources are not present in local VO, then regular node *p* forwards *q* to a selected SuperNode *m* (represented by SN_m) in local VO and *m* will multicast *q* to SuperNodes in other VOs according to its SNList.

Rule 4. $\forall SN_m(((SN_m \in PS_{\text{local}}) \wedge ((Receive(q)) \vee (ZS_{\text{local}} = \varnothing))) \Rightarrow \exists \text{SNList}\ (MultiCast(q, \text{SNList})))$.

Rule 4 indicates that SuperNode m multicasts q to the SuperNodes in other VOs according to its SNList (actually GlobalNeighborList introduced in 3.4).

Rule 5. $\exists p \exists PS_i((p \in PS_i) \wedge (PS_i \subseteq \mathrm{PS}) \wedge (MatchValue_{(p,q)} = 1)) \Rightarrow \exists ZS_{global}(Return(ZS_{global}) \wedge Exit)$.

If the desired resources are located in global scope, then the discovery process will terminate and ZS_{global} are returned to the node initiated the request.

Rule 6. $\forall p \forall PS_i(((p \in PS_i) \wedge (PS_i \subseteq \mathrm{PS}) \wedge (MatchValue_{(p,q)} = 0)) \Rightarrow ((ZS_{global} = \varnothing) \wedge Exit))$.

If the desired resources are not present in grid, then the discovery process will terminate and NULL will be returned to the node initiated the request. The above six rules and Fig. 2 (a) could describe the situations with regard to regular nodes in discovery process of SDRD. The rules describing the SuperNodes′ match mechanism is shown in Fig. 2 (b).

3.4 Organization of SDRD Overlay

Nodes in SDRD are organized according to geographical proximity or physical topology. Nearby nodes cluster into one VO. This could alleviate the mismatch problem between P2P overlay network and the physical underlying network mentioned in section 2.

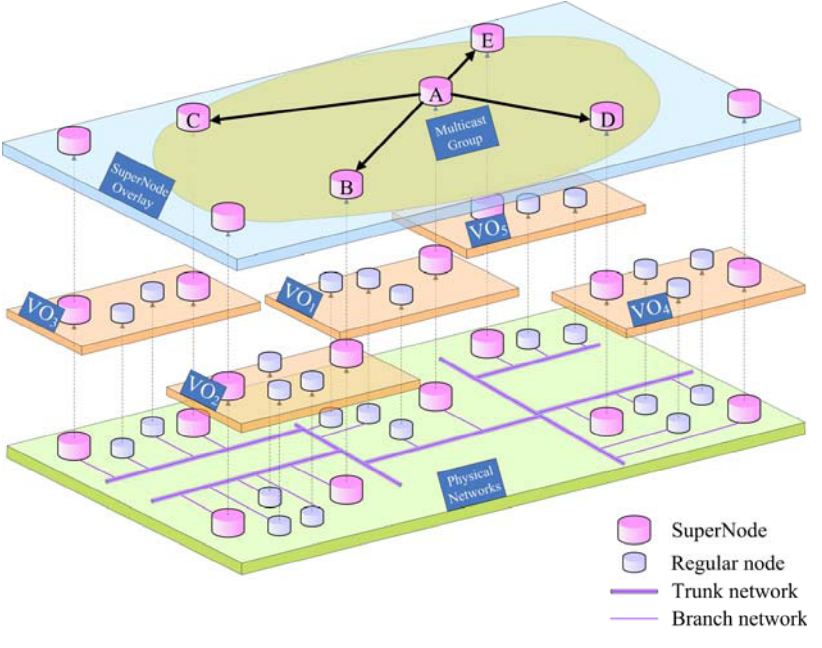


Fig. 3. Example of SDRD overlay

All the VOs form the middle layer of SDRD overlay, see Fig. 3. Each VO contains multiple SuperNodes. All the SuperNodes form the top layer of SDRD overlay. Each VO employs one of the improved DHTs. Every regular node maintains a list called LocalSNList, which contains IDs of SuperNodes in the same VO. Besides that, a SuperNode maintains a list called GlobalNeighborList containing IDs of SuperNodes in other VOs, and only one ID is needed per every other VO, for exmaple, in Fig. 3, GlobalNeighborList of $SuperNode_A$ contains $SuperNode_B$, $SuperNode_C$, $SuperNode_D$ and $SuperNode_E$, which is represented by bold lines.

SDRD employs improved DHTs to locate resources in VOs due to the following reasons.

First, DHTs can eliminate a series of problems generated by using centralized servers, such as performance bottleneck, a single point of failure.

Second, grids are less dynamic than P2P networks [7]. This feature prevents grids from churn [16] in DHTs.

Third, compared to unstructured P2P networks, DHTs guarantee that the resources will always be found if it is available in local VO.

Fourth, routing efficiencies of DHTs are high, thus, DHTs can avoid the network overhead caused by flooding in unstructured P2P like Gnutella.

In addition, each VO can choose one of the improved DHTs freely, which brings flexibility to routing in SDRD.

4 Key Technologies

4.1 SuperNodes in SDRD

Super nodes of related works (the term ″super nodes″ refer to the ones in other works) [1], [7], [15] are in charge of two tasks. First, they have to collect and index grid resource information of each VO, and respond to regular nodes′ requests. Second, if the desired resources are not present in local VO, super nodes forward the requests to other super nodes in different VOs. This strategy might have two principal drawbacks. One is that the super nodes might become potential bottlenecks. The other is that all of the information about resource characteristics (such as CPU load) will be sent to super nodes when they are changed, which will cause routing hotspot in networks.

Without the help of super nodes, VOs in SDRD use the improved DHTs to route the request messages in local VO. Therefore, there is only one task for SuperNode to do, which is forwarding requests to other SuperNodes in different VOs when the desired resources are not present in local VO (according to Rule 4 in 3.3). This strategy will eliminate the drawbacks of related works. The scalability of SDRD will benefit from the lower load on SuperNodes.

In SDRD, SuperNodes are selected based on the criterions below.

- evaluation of a node's service capabilities, including the node's computation resources, storage resources and network bandwidth.
- evaluation of a node's availability, e.g. mean time to failure.
- evaluation of a node's average load.

Generally speaking, nodes with high service capabilities, high availability and low average load will be selected to SuperNodes.

4.2 Routing APIs

Routing APIs to implement the rules described in 3.3 are introduced in this subsection. The API used to route requests by a regular node is as follows. Invoked procedures are depicted by bold type.

```
//Procedure name: regular_node_discovery(q)
call local_DHT(alg, q, LocalResultSet);
if (LocalResultSet != 0)
  call rank(LocalResultSet);
  return LocalResultSet;
else
  call local_SuperNode(m, q);
  if(GlobalResultSet !=0)
    call rank(GlobalResultSet);
    return GlobalResultSet;
  else return NULL;
```

If SuperNode *m* initiated the request *q* itself, the routing API will be the next one.

```
//Procedure name: super_node_discovery(q)
call local_DHT(alg, q, LocalResultSet);
if (LocalResultSet != 0)
  call rank(LocalResultSet);
  return LocalResultSet;
else
  call multicast(GlobalNeighborList, q);
  k = sizeof(GlobalNeighborList);
  for i=1,…,k
    call merge_results(SuperNode_i's LocalResultSet,
GlobalResultSet);
  if(GlobalResultSet !=0)
    call rank(GlobalResultSet);
    return GlobalResultSet;
  else return NULL;
```

Significant difference of discovery procedure between SuperNode and regular node is that there is no need for SuperNode to forward the request to other SuperNodes in the same VO (recall Fig. 2). SuperNode *m* multicasts the request *q* to the SuperNodes present in *m*′s GlobalNeighborList.

4.3 SuperNode Multicast

As an example, we will show that how SuperNode *m* constructs its multicast tree. Note that, *m*′s Multicast Group = {*m*, *m*′s GlobalNeighborList}. According to the network latency, *m* selects SuperNode neighbours in different VOs. After that, *m* will construct a multicast tree. Constructing a multicast tree is a Steiner tree problem in

networks, and several heuristic algorithms could be used to solve this problem, such as the MPH algorithm [17].

5 Analysis

5.1 Routing Efficiency

Routing efficiency of SDRD is determined by several factors: (1) improved DHTs used by local VO, i.e. *Alg* in Rule 1; (2) overhead caused by multicasting request messages to other SuperNodes in different VOs; (3) improved DHTs used by other VOs. Suppose that, compared to the time consumed by network transport, the time consumed by computer process can be omitted. If node p issues a request, and the routing process exits with Rule 2, time consumption will be $T_{SDRD} = T_{VOlocal}$, where $T_{VOlocal}$ represents the time consumed by improved DHTs in local VO. If the routing process exits with Rule 5 or Rule 6, then time consumption will be as follows: $T_{SDRD} = T_{VOlocal} + \max\{T_{VOi} + T_{<m,i>}\}$, where $T_{<m,i>}$ represents time consumed by multicasting, and $m \in p'$s LocalSNList, $i \in m'$s GlobalNeighborList.

5.2 Apply to Other Models

Table 1 summarizes the models that could adopt SDRD for resource discovery, the last row of Table 1 shows which part of SDRD is corresponding to the related part in other models. SDRD could reduce load on super-peers. As a result, the scalability of these models will be improved due to the lower load on SuperNodes.

Table 1. Models could adopt SDRD

Model	Super node	Organization Unit
constellation model [1]	fixed star	solar system
super-peer model [7]	super-peer	VO／cluster
P2P based GIS [15]	super-peer	cluster
SDRD	SuperNode	VO

6 Experiments

The test bed is the Xi′an Node of the China National Grid, located in Xi′an Jiaotong University.

6.1 Evaluation of Routing Efficiency and Locating Guarantee in VO

Like [2], we carried out this experiment with simple requests (1-dimensional request) and perfect matching. We have compared improved DHTs (MAAN [10]) with random flooding strategies in [2]. The experimental results are shown in Fig. 4.

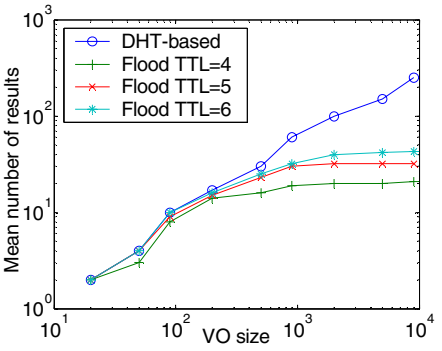


Fig. 4. Evaluation of routing efficiency and locating guarantees in VO

TTL (Time-To-Live) of flooding strategies is set to 4, 5 and 6 (according to reference [7]), respectively. Fig. 4 shows that the flooding strategies are less efficient than the improved DHTs and provide no locating guarantees.

6.2 Evaluation of the Load on SuperNode

We tested SuperNode throughputs to verify if SDRD could decrease the load on super nodes in other models. This experiment is carried out under the framework of the constellation model [1]. Number of requests processed by the SuperNode per minute is shown in Fig. 5. With the help of SDRD, SuperNode (fixed star) could process much more requests than that of the original approach used by the constellation model.

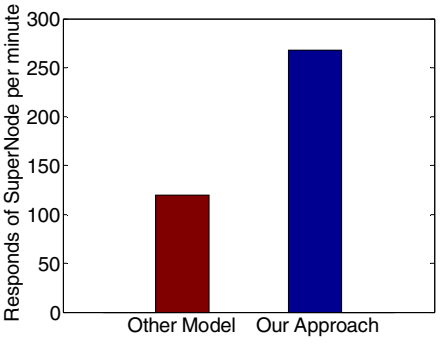


Fig. 5. Evaluation of the load on SuperNode

6.3 Evaluation of Overall Routing Efficiency

Evaluation results of overall routing efficiency for SDRD are depicted in Fig. 6. We assumed that mean size of VO is 50, there are five types of resources, and resources obey balanced distribution. Four groups of results were present in Fig. 6.

Test 1 represents the simple requests and perfect matching, the response time increases with the multicast group size. Test 2, test 3 and test 4 shows the response time with regard to 1-d (1-dimensional) range query, 2-d range query, 5-d range query, respectively. As shown in Fig. 6, we could see that after the multicast group size reaches a certain value, for example, 0.9 second with regard to test 3, the response time increases very slowly and maintains at a certain level even the requests are sent to more VOs. Therefore, the scalabilities of the models adopting our approach could be enhanced by this characteristic of SDRD.

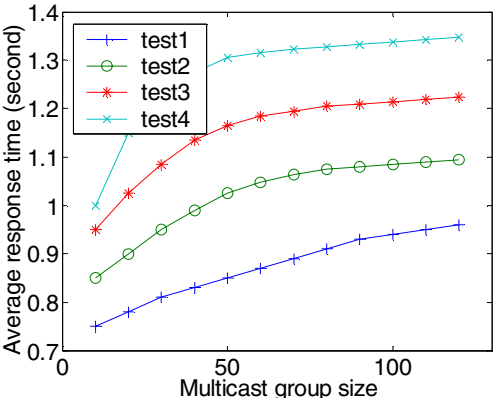


Fig. 6. Overall routing efficiency evaluation

7 Conclusion

We propose SDRD, a SuperNode and DHT based approach to resource discovery in grid environments. Sophisticated routing strategies are implemented to improve the scalabilities and routing efficiencies. Experimental results show that SDRD scales well and offers an efficient decentralized discovery service in grid.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant No. 60773118), 863 Project of China (Grant No. 2006AA01A109) and Program for Changjiang Scholars and Innovative Research Team in University.

References

1. Wang, Y., Dong, X., He, X., et al.: A Constellation Model for Grid Resource Management. In: Cao, J., Nejdl, W., Xu, M. (eds.) APPT 2005. LNCS, vol. 3756, pp. 263–272. Springer, Heidelberg (2005)
2. Iamnitchi, A., Foster, I.: On Fully Decentralized Resource Discovery in Grid Environments. In: Lee, C.A. (ed.) GRID 2001. LNCS, vol. 2242, pp. 51–62. Springer, Heidelberg (2001)

3. Androutsellis-Theotokis, S., Spinellis, D.: A Survey of Peer-to-Peer Content Distribution Technologies. ACM Computing Surveys 36(4), 335–371 (2004)
4. Talia, D., Trunfio, P.: Toward a Synergy Between P2P and Grids. IEEE Internet Computing, pp. 96, 94–95 (July/August 2003)
5. Abdullah, A., Othman, M., Sulaiman, M.N., et al.: Data Discovery Mechanism for a Large Peer-to-Peer Based Scientific Data Grid Environment. In: Laganà, A., Gavrilova, M., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3044, pp. 146–157. Springer, Heidelberg (2004)
6. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. J. Comput. Sci & Technol. 21(4), 513–520 (2006)
7. Mastroianni, C., Talia, D., Verta, O.: A Super-Peer Model for Building Resource Discovery Services in Grids: Design and Simulation Analysis. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 132–143. Springer, Heidelberg (2005)
8. Ranjan, R., Harwood, A., Buyya, R.: A Study on Peer-to-Peer Based Discovery of Grid Resource Information, http://www.gridbus.org/reports/pgrid.pdf
9. Ganesan, P., Yang, B., Garcia-Molina, H.: One Torus to Rule them All: Multi-dimensional Queries in P2P Systems. In: WebDB 2004, pp. 19–24. ACM Press, New York (2004)
10. Cai, M., Frank, M., Chen, J., et al.: MAAN: A Multi-Attribute Addressable Network for Grid Information Services. Journal of Grid Computing 2(1), 3–14 (2004)
11. Andrzejak, A., Xu, Z.: Scalable, efficient range queries for grid information services. In: IEEE P2P 2002, pp. 33–40. IEEE Press, Los Alamitos (2002)
12. Schmidt, C., Parashar, M.: A Peer-to-Peer Approach to Web Service Discovery. World Wide Web: Internet and Web Information Systems 7(2), 211–229 (2004)
13. Tam, D., Azimi, R., Jacobsen, H.A.: Building Content-Based Publish/Subscribe Systems with Distributed Hash Tables. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) Databases, Information Systems, and Peer-to-Peer Computing. LNCS, vol. 2944, pp. 138–152. Springer, Heidelberg (2004)
14. Crainiceanu, A., Linga, P., Gehrke, J., Shanmugasundaram, J.: Querying Peer-to-Peer Networks Using P-Trees. In: WebDB 2004, pp. 25–30. ACM Press, New York (2004)
15. Puppin, D., Moncelli, S., Baraglia, R., et al.: A Grid Information Service Based on Peer-to-Peer. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 454–464. Springer, Heidelberg (2005)
16. Rhea, S., Geels, D., Roscoe, T., et al.: Handling Churn in a DHT. In: USENIX Annual Technical Conference, Boston, MA, USA, pp. 127–140 (2004)
17. Takahashi, H., Matsuyama, A.: An Approximate Solution for the Steiner Problem in Graphs. Math Japonica 24, 573–577 (1980)
18. Liu, Y., Xiao, L., Liu, X., et al.: Location Awareness in Unstructured Peer-to-Peer Systems. IEEE Transactions on Parallel and Distributed Systems (TPDS) 16(2), 163–174 (2005)
19. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley/Pearson Press, London (1999)

A Comparative Study of Two Java High Performance Environments for Implementing Parallel Iterative Methods

Jacques M. Bahi, Raphaël Couturier, David Laiymani,
and Kamel Mazouzi

Laboratoire d'Informatique de l'université de Franche-Comté (LIFC)
IUT de Belfort-Montbéliard - Rue Engel Gros
BP 527 90016 Belfort CEDEX - France
name@iut-bm.univ-fcomte.fr

Abstract. This paper aims at studying two Java high performance environments in order to implement parallel iterative methods on Grid infrastructures. We exhibit the important features offered by MPJ Express and Jace V2 to tackle the different issues linked to parallel iterative algorithms. Our study relies on the implementation of a typical iterative application: the multi-splitting method on a large scale grid platform.

1 Introduction

Currently there is a growing interest in developing Grid applications using the Java language. Even if its performance are not comparable to those of the C language for example, many reasons could explain this interest. Among them we can quote the two following ones. First, the ability of Java to handle the heterogeneity between different hardware and operating systems. With regard to the highly heterogeneous nature of the Grid this feature is essential. Second, its ability to support efficient communication. Becker et al shows in [1] how the Java NIO API [2] provides scalable non-blocking I/Os which perform very well.

It also appears that most Grid parallel applications use the message-passing paradigm. In this model, tasks co-operate by exchanging messages and the Message Passing Interface (MPI) is a standard for implementing message-passing applications. In this context, and considering the advantages of Java previously exposed, it is not surprising to see that many research projects aim at developing a message-passing system in Java. These projects can be classified into three classes. The projects of the first class [3] are built upon JNI [4] and use a native MPI implementation as communication layer. These projects provide efficient communication procedures but are not "pure" Java. Projects of the second class [5,6] are based on Java RMI. The RMI API is an elegant high-level "pure" Java solution for remote method invocations of distributed objects but offers little communication performances. In the third class, projects [1,7] use a low-level approach based on Java sockets. This ensures good communication performances and a truly "pure" Java portable environment.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 313–321, 2007.

Now, from an application point of view it appears that the well-known parallel iterative numerical algorithms have been the main class used in scientific applications so far. Unfortunately, they usually require several inter-processor communications and synchronizations (to update data and to start the next computation steps for example). The aim of this paper is to provide a comparative study of two "pure" Java message-passing environments for implementing parallel iterative numerical algorithms. We focus on MPJ Express which is developed at the University of Reading [1] and on Jace an environment we are currently developing at the Université de Franche-Comté. We present the features offered by these two environments to tackle the different issues linked to parallel iterative applications. Both of them use the Java NIO socket API and Jace also allows to implement a particular iterative model called *AIACs* (*Asynchronous Iteration-Asynchronous Communication*) algorithms [8,9]. Our study relies on the implementation, on the Grid'5000 testbed [10], of a typical numerical iterative method: the multisplitting method. We show that the communication layer of MPJ offers better performances than Jace. Nevertheless, the ability of Jace V2 to build asynchronous implementations allows it to outperform MPJ synchronous implementations.

This paper is organised as follows. In section 2, we present the motivations and scientific context of our work. In section 3, we describe the MPJE and Jace environments. We particularly describe the new architecture and the new features that we have implemented in the Jace environment (named Jace V2). Section 4 details the experiments we conducted on the Grid'5000 testbed. We present the multisplitting method (both its synchronous and asynchronous implementations) and we analyze the results of our test. We end in section 5 by some concluding remarks and future work.

2 Scientific Context and Motivations

As exposed in the introduction, parallel iterative methods are now widely used in many scientific domains. In the same way and due, in part, to its ability to tackle the heterogeneity problem of the Grid, the Java language is now a good candidate for developping high performance applications. But what are the main features that a Java programming environment must offer to develop efficient numerical iterative applications ?

Parallel iterative algorithms can be classified in three main classes depending on how iterations and communications are managed (for more details readers can refer to [11]). In the *Synchronous Iterations - Synchronous Communications (SISC)* model data are exchanged at the end of each iteration. All the processors must begin the same iteration at the same time and important idle times on processors are generated. The *Synchronous Iterations - Asynchronous Communications (SIAC)* model can be compared to the previous one except that data required on another processor are sent asynchronously i.e. without stopping current computations. This technique allows to partially overlap communications by computations but unfortunately, the overlapping is only partial and important idle times remain. It is clear that, in a grid computing context, where computational nodes are large, heterogeneous and widely distributed, the idle times generated by synchronizations are very penalizing. One way to overcome this problem is to use the *Asynchronous Iterations - Asynchronous Communications (AIAC)* model. Here, local computations do not need to wait for required data. Processors can then perform

their iterations with the data present at that time. Figure 1 illustrates this model where the grey blocks represent the computation phases, the white spaces the idle times and the arrows the communications. With this algorithmic model, the number of iterations required before the convergence is generally greater than for the two former classes. But, and as detailed in [8], AIAC algorithms can significantly reduce overall execution times by suppressing idle times due to synchronizations especially in a grid computing context.

In this context, it appears that communications and synchronizations are crucial points that any message-passing environment must managed carefully. This communication management must be based on: an efficient point-to-point communication module, an efficient thread management module (for scalability reasons) and the ability to easily implement AIAC algorithms.

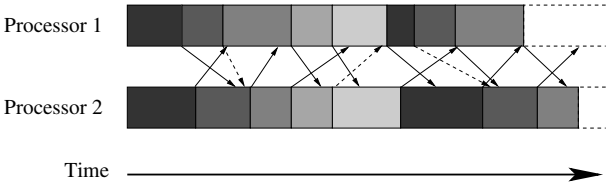


Fig. 1. The Asynchronous Iterations - Asynchronous Communications model

In the remainder, we present MPJ Express and Jace, two Java message-passing environments which aim at offering these features.

3 The MPJ Express and Jace V2 Environments

3.1 MPJ Express

MPJE is structured into a layered design (see figure 2) which allows to use different communication devices such as NIO or native MPI via JNI. In the following we only focus on the NIO device driver. In [1] Baker et al show how point-to-point NIO communications perform well and in [12] Pugh et al point out the good scalability of this package with respect to standard Java sockets.

The communication protocols. The NIO device driver (called `mjdev`) proposes three communication protocols.

- *The Eager-Send protocol*. This protocol is used for small messages (size smaller than 128 Kbytes). Assuming that the receiving part has got an unlimited memory for storing messages the number of control messages is minimized.
- *The Rendezvous protocol*. This protocol is used for large messages (size greater than 128 Kbytes). In this case, control messages are exchanged since their overhead is negligible.
- *The Shared Memory protocol*. This protocol is used when a process is sending a message to itself (inside the same JVM).

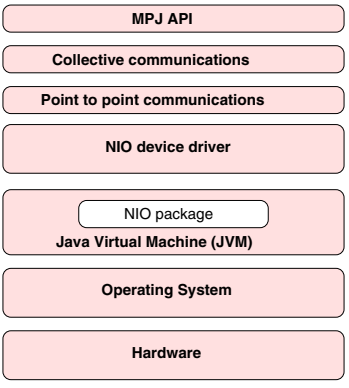


Fig. 2. The MPJ layered architecture

The buffering API. Java sockets are not able to directly access memory in order to read/write basic datatypes. Furthermore, the lack of pointers management could make difficult the use of complex operations such as gather/scatter. To overcome these difficulties, MPJ provides a buffering API in order to pack and unpack data to be sent [13]. Two kinds of buffers can be used: *static* and *dynamic* buffers. Static buffers can only contain primitive datatypes while dynamic buffers can deal with serialized Java objects. When a buffer is created, read and write operations are available to pack and unpack data on it.

The communications primitives. Blocking and non-blocking sending methods are available. These methods are called in the user thread and use the communication protocols previously described. In the same way, blocking and non-blocking receiving methods are available. These methods can be initiated by the user thread (eager-send protocol) or by the NIO selector thread (rendez-vous protocol).

Advantages and drawbacks. The MPJ API is complete and offers a MPI-like style of programming which makes the porting of existing applications easier. Based on a solid buffer management its communication layer is efficient and MPJ appears to be standard for implementing Java message-passing applications. Nevertheless, it appears that MPJ is not well suited for AIAC algorithms. Indeed, even if it is a thread-safe environment, its communication layer architecture is mono-threaded. It is shown in [8] that with this kind of process management it is difficult to implement efficient AIAC algorithms. Another drawback of MPJ is that the application deployment procedure can suffer from a lack of scalability since centralized communication schemes are used.

3.2 Jace V2

Jace [6] is a Java programming and executing environment that permits to implement efficient asynchronous algorithms as simply as possible. Jace builds a distributed virtual machine, composed of heterogeneous machines scattered over several distant sites. It proposes a simple programming interface to implement applications using the message

passing model. The interface completely hides the mechanisms related to asynchronism, especially the communication management and the global convergence control. In order to propose a more generic environment, Jace also provides primitives to implement synchronous algorithms and a simple mechanism to swap from one mode to another. Jace relies on four components: *the daemon, the worker, the computing task* and *the spawner*.

The daemon. The daemon is the core of the Jace system, it is launched on each node taking part in the computation. When a daemon is launched, a remote server is started on it and continuously waits for remote invocations. This server provides communications between the daemons and the spawner. It is used to manage the Jace environment like for example: initializing the workers, monitoring and gathering the results . . . Daemons are structured as a binomial tree. This hierarchical view of the machines set achieves more efficient spawning and optimizes global communications.

The worker. The worker is the entity responsible for executing user applications. It is a Jace service created for each execution by the daemon. Figure 3 shows the internal architecture of the worker which is composed of two layers:

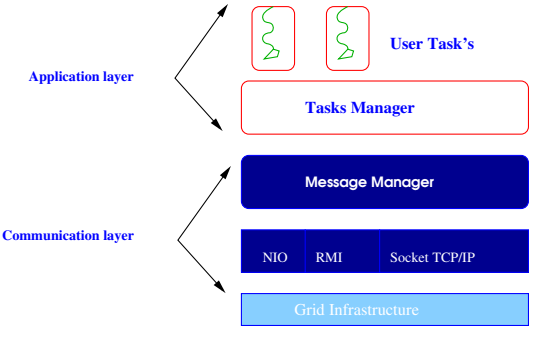


Fig. 3. Jace worker architecture

- **The Application Layer.** This layer provides tasks execution and global convergence detection. A daemon may execute multiple tasks, allowing to reduce distant communications. Jace is designed to control the global convergence process in a transparent way. Tasks only compute their local convergence state and call the Jace API to retrieve the global state. The internal mechanisms of the convergence detection depend on the execution mode i.e. synchronous or asynchronous.
- **The Communication Layer.** Communications between tasks are performed using the message/object passing model. Jace uses waiting queues to store incoming/outgoing messages and two threads (`sender` and `receiver`) to deal with communications. According to the kind of algorithm used, synchronous or asynchronous, queues managements are different. For a synchronous execution, all messages sent by a task must be received by the other tasks. Whereas on an asynchronous execution, only the most recent occurrence of a message, with the same

source or destination and containing the same type of information, is kept in the queues. The older one, if existing, is deleted. For scalability issues and to achieve better performances, the communication layer should use an efficient protocol to exchange data between remote tasks. For this reason Jace is based on several protocols : TCP/IP Sockets, NIO (New Input/Output) [2,12] and RMI (Remote Method Invocation).

The Computing Task. As in MPI-like environments, the programmer decomposes the problem to be solved into a set of cooperating sequential tasks. These tasks are executed on the available processors and invoke special routines to send or receive messages. A `task` is the computing unit in Jace, which is executed like a thread rather than a process. Thus, multiple tasks may be executed in the same worker and can share system resources.

We also point out here that Jace implementation relies on the Java object serialization to transparently send objects rather than raw data.

The Spawner. The spawner is the entity that effectively starts the user application. After starting daemons on all nodes, computations begin by launching the spawner program with some parameters (the number of tasks to be executed, the URL of the task byte-code, the parameters of the application, the list of target daemons, the mapping algorithm (round robin, best effort)). Then, the spawner broadcasts this information to all the daemons. For scalability reasons, that is achieved by using an efficient broadcast algorithm based on a binomial tree [14]. When a daemon receives the spawner message, it forwards this information to its neighbors and starts a worker to load and execute the user tasks.

Advantages and drawbacks. As presented above Jace is a multi-threaded environment very suitable for AIAC algorithms. For scalability reasons its application deployment procedure is designed in a highly distributed way. Unlike the MPJ one, the Jace communication layer is able to transfer any kind of data objects. This feature provides more flexibility but requires objects serialization which decreases overall performances.

4 Experiments

4.1 The Application: The Multisplitting Method

Consider the n dimensional linear system: $Ax = b$. As exposed in figure 5, the A matrix is split into horizontal rectangle parts. Each of these parts is then affected to one processor. With this distribution, a processor is in charge of computing its $XSub$ part by iteratively solving the following subsystem:

$$ASub * XSub = BSub - DepLeft * XLeft - DepRight * XRight$$

This resolution can be processed by a direct solver such as SuperLu. Then the solution $XSub$ must be sent to each processor which depends on it. Now, if rectangle matrices are not disjoint, it appears that some computations will be redundant. This property

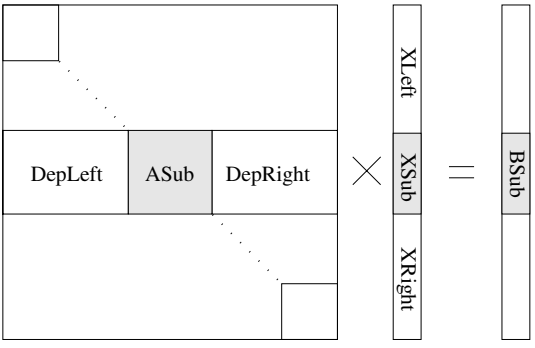


Fig. 4. Decomposition of the system

is called *overlapping* and should be taken into account. In this way, three policies can be applied. Either a processor ignores its components if its neighbors has computed them, or it ignores the neighbors components or it mixes the shared components (by computing their average for example). This parameter has an influence on the convergence speed of the algorithm. Interesting reader can find more details in [15].

For our purpose, this application is interesting for many reasons. First, the convergence of the method for both synchronous and asynchronous mode is shown in [15] (with some restrictions on the A matrix). Second, it appears that the computation/communication ratio does not let performances be too dependant on the communication layer. In this way, we must be able to evaluate the whole of the target environments (memory management, threads management . . .). Finally, this application is not a "toy" application. It covers several scientific computation areas and its study in different contexts is relevant.

4.2 Experiments Results

The experiments have been conducted on the Grid'5000 platform. This testbed is composed of an average of $1,300$ bi-processors that are located in 9 sites in France: Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis and Toulouse. The inter-sites links range from 2.5Gbps up to 10Gbps while most of the sites have a Gigabit Ethernet Network for local machines. For more details on the Grid'5000 architecture, interested readers can refer to: www.grid5000.fr. All the nodes run the Linux Debian distribution with the Sun Java 1.5 Java Virtual Machine.

Figure 5 shows the execution times of the multi-splitting application for different matrix sizes and with 150 nodes of the Grid'5000 testbed. The processors were distributed over 2 sites and the two Jace implementations (synchronous and asynchronous) rely on the Socket communication layer.

It appears that the two synchronous versions (MPJ and Jace) present some equivalent execution times. With respect to the architecture of the two communication layers, these results can be surprising since no object serialization is performed with MPJ while Jace requires this kind of process. We can explain these results by the fact that the target application is coarse grain and so is less sensitive to communication performances. These

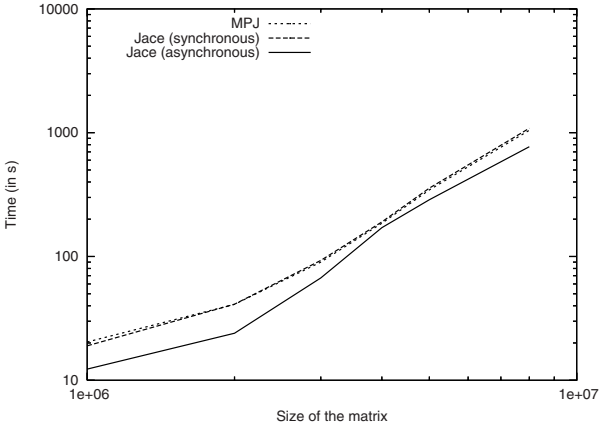


Fig. 5. Time to solve different generated matrices

tests also show the interest of AIAC algorithms since the Jace asynchronous version clearly outperforms the two synchronous ones. The fundamental properties of AIACs algorithms can explained these results. In particular, we can see here the efficiency of asynchronism which allows to obtain a good computations/communications overlapping. This underlines an important feature for Java high performance environments: the ability of easily implemented AIAC algorithms.

5 Concluding Remarks and Future Work

In this paper, we have presented a comparative study of two Java high performance environments (MPJ Express and Jace V2) for implementing parallel iterative methods. Through the implementation of a typical iterative application (the multi-splitting method) we have shown that the communication layer of MPJ Express is efficient. We have also shown that the ability of Jace to support the AIAC model is very relevant.

We are currently working on how AIAC algorithms behave on a peer-to-peer (P2P) architecture. We study the integration of Jace on P2P environments such as JXTA or ProActive [16] since these environments already propose standard P2P services such as failure detection, NAT traversing.

References

1. Baker, M., Carpenter, B., Shafi, A.: MPJ Express: Towards Thread Safe Java HPC. In: Cluster Computing, Barcelona, sept 2006, IEEE Computer Society Press, Los Alamitos (2006)
2. New I/O API, http://java.sun.com/j2se/1.4.2/docs/guide/nio
3. Ma, R., Wang, C.-L., Lau, F.: M-javampi: A java-mpi binding with process migration support. In: CCGRID 2002. Proc. of the 2nd IEEE/ACM Int. Symposium on Cluster Computing and the Grid, p. 255. IEEE Computer Society Press, Los Alamitos (2002)
4. JNI, http://java.sun.com/j2se/1.4.2/docs/guide/jni/

5. Morin, S., Koren, I., Krishna, C.M.: Jmpi: Implementing the message passing standard in java. In: IPDPS 2002. Proc. of the 16th Int. Parallel and Distributed Processing Symposium, p. 191. IEEE Computer Society Press, Los Alamitos (2002)
6. Bahi, J., Domas, S., Mazouzi, K.: Jace: a java environment for distributed asynchronous iterative computations. In: 12th Euromicro Conference PDP 2004, pp. 350–357. IEEE Computer Society Press, Los Alamitos (2004)
7. MPP, http://www.uib.no/People/nmabh/mtj/mpp/
8. Bahi, J., Contassot-Vivier, S., Couturier, R.: Performance comparison of parallel programming environments for implementing AIAC algorithms. Journal of Supercomputing 35(3), 227–244 (2006)
9. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice Hall, Englewood Cliffs NJ (1989)
10. Grid'5000, http://www.grid5000.fr
11. Bahi, J., Contassot-Vivier, S., Couturier, R.: Asynchronism for iterative algorithms in global computing environment. In: 16th Int. Symposium on High Performance Computing Systems and Applications, Moncton, Canada, pp. 90–97. IEEE Computer Society Press, Los Alamitos (2002)
12. Pugh, B., Spaccol, J.: MPJava: High Performance Message Passing in Java using Java.nio. In: Proceedings of the Workshop on Languages and Compilers for Parallel Computing, College Station, Texas, USA (October 2003)
13. Baker, M., Carpenter, B., Shafi, A.: An Approach to Buffer Management in Java HPC Messaging. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J. (eds.) ICCS 2006. LNCS, vol. 3991, Springer, Heidelberg (2006)
14. Gerbessiotis, A.V.: Architecture independent parallel binomial tree option price valuations. Parallel Computing 30(2), 301–316 (2004)
15. Bahi, J.M., Couturier, R.: Parallelization of direct algorithms using multisplitting methods in grid environments. In: IPDPS 2005, pp. 254b, 8 pages. IEEE Computer Society Press, Los Alamitos (2005)
16. Caromel, D., Di Constanzo, A., Mathieu, C.: Peer-to-peer for computational grids: Mixing clusters and desktop machines. Parallel Computing (2007)

SIGRE – An Autonomic Spatial Information Grid Runtime Environment for Geo-computation

ZhenChun Huang[1], GuoQing Li[2], Bin Du[1], Yi Zeng[2], and Lei Gu[1]

[1] Department of Computer Science and Engineering, Tsinghua University, Beijing 100084
[2] China Remote Sensing Satellite Ground Station, Beijing 100086
huangzc@tsinghua.edu.cn, gqli@ne.rsgs.ac.cn,
dubin@mails.tsinghua.edu.cn, yzeng@ne.rsgs.ac.cn,
jackflit98@mails.tsinghua.edu.cn

Abstract. Spatial Information Grid is a kind of application grid which tries to connect resources such as computer, data sources, and processing algorithms, and builds a distributed, robust, flexible and powerful infrastructure for geo-computation. It needs a powerful and easy-to-use running environment. In this paper, an autonomic runtime environment for geo-computation is proposed and named SIGRE — the Spatial Information Grid Runtime Environment. Based on it, SIG resources can be distributed, discovered, and matched autonomically. And a distributed, flexible and powerful data infrastructure which can distribute data with different types, different sources, and different goals in a uniform interface easily and flexibly is founded. Based on the implementation of SIGRE by java language, a SIG testbed is constructed, and the test on it shows that SIGRE can provide a powerful, easy-to-use, robust and autonomic runtime environment for SIG, and developers can develop SIG resources and SIG applications on SIGRE easily and quickly.

Keywords: geo-computation, spatial information grid, runtime environment.

1 Introduction

Geo-computation can be regarded as the application of the computational science paradigm to study a wide range of problems in geographical and Earth science context. It is concerned with new computational techniques, algorithms, and paradigms that are dependent upon and can take advantage of techniques such as Grid computing, high performance computing and high throughput computer. Based on grid computing technologies, the Spatial Information Grid (shortly SIG) tries to connect resources such as high performance computers, high throughput data storages and spatial data processing algorithms; enable the sharing, co-operation and collaboration among the resources and their consumers; and build a distributed, robust, flexible, and powerful infrastructure for geo-computation. Based on SIG, resources such as computing powers, spatial information data, and processing models and algorithms can be shared; and distributed geo-computation applications can be built easily and quickly.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 322–329, 2007.

In brief, SIG can be regarded as a kind of application grid for the geo-computation researchers. It is based on the general grid infrastructure; and built by the general grid infrastructure and dozens of SIG extensions and improvements, such as the SIG job framework, SIG data infrastructure, SIG registry meta-service extension, and so on. In SIG, the general grid infrastructure makes it possible to share resources and co-operating among resources. On the other hand, the SIG extensions and improvements make the infrastructure more powerful, friendly, and adaptive for the geo-computation users, and make it easier and quicker to develop SIG applications on the infrastructure.

As viewed from the physical deployment, SIG is constructed by a lot of distributed SIG nodes which are connected by network (usually Internet). All resources in SIG are attached to the SIG nodes, and accessed through the nodes. At the same time, most of the web-based SIG applications are deployed on the SIG nodes and provide web UI for their users, too. Whether the SIG resources or the SIG applications need a basic support running environment, which will be proposed later and named SIGRE — the Spatial Information Grid Runtime Environment, an autonomic runtime environment for geo-computation.

Most traditionally, the "runtime environment" is implemented as a set of support libraries on the target platform for the basic functions of the applications. These libraries are invoked by the application codes through a well-defined API (Application Programming Interface) and serve applications the most frequently used functions. One of the most famous samples is the Unix lib5 runtime environment for C applications, which carries the basic functions such as file accessing and process management. And there are different platform-dependent lib5 libraries on different platforms, and they support the same functions. The Win32 runtime environment is another sample for this set.

By the growing up of virtual machine techniques, virtual machine is adopted as a part of the runtime environment, and provides a platform-independent running core for applications; so that the applications may move from one platform to another smoothly and quickly without modification. It can be called "VM based runtime environment". One of the most famous VM based runtime environment is java runtime environment which is often called JRE. In JRE, a virtual machine named JVM (Java Virtual Machine) is included, and interprets the java virtual codes for the java applications. And at the same time, a set of java runtime libraries are included to provide frequent functions such as iostream or url supports. The JRE supports different platforms, and makes it possible to "build one, run anywhere". The .Net framework is also another sample of this kind of runtime environments.

Though the VM based runtime environment provides a platform-independent environment for applications by a well-defined virtual machine and a set of support libraries, it is still not enough for the applications today, especially for the resources and applications in grid such as SIG. To develop, deploy and manage resources and applications based on a VM based runtime environment is still a hard work. So, an autonomic runtime environment which is able to self-configure, self-manage, self-heal and self-optimize is needed for better development and execution of the resources and applications.

In this paper, based on the techniques including web service, grid computing, and spatial information grid, an autonomic runtime environment named SIGRE (Spatial

Information Grid Runtime Environment) is proposed and implemented for geo-computation. Each SIG node carries out an instance of SIGRE, which provides the basic running environment including the virtual machine, runtime libraries, management and optimization components, monitoring and healing components, etc. All SIGRE instances on the SIG nodes makeup an environment for the execution and management of resources and applications in SIG, and provide the key functions including resource discovering, data managing, service quality supporting, etc.

2 Architecture

There are two connotations of SIGRE: the SIGRE instance deployed on a SIG node, which is called SIGRE instance; and the runtime environment built by all the SIGRE instances on SIG nodes, which is called SIGRE platform. As viewed from the logical architecture, SIGRE is built up by the following components: Java virtual machine and runtime environment; general web server and servlet container; SIG resource and job framework; SIG data infrastructure support; SIG resource registry and discovery; SIG service and node monitor; SIG service quality control and management; and so on. It is shown in Figure. 1.

SIGRE is founded based on the following components: JVM and JRE [1], web server, and servlet container [2]; and they should be downloaded and installed separately for the sake of the user's free choice on them. SIG resource and job framework [3] provides a basic job framework and make it possible to create, run, monitor, and manage an SIG job through web service interface. Most of the SIG resources are implemented and provided as SIG jobs. With the toolkits provided by SIGRE, it is very simple and easy to develop and deploy an SIG job so that almost each user who knows a little of Java can share his resources by develops and deploys new SIG jobs.

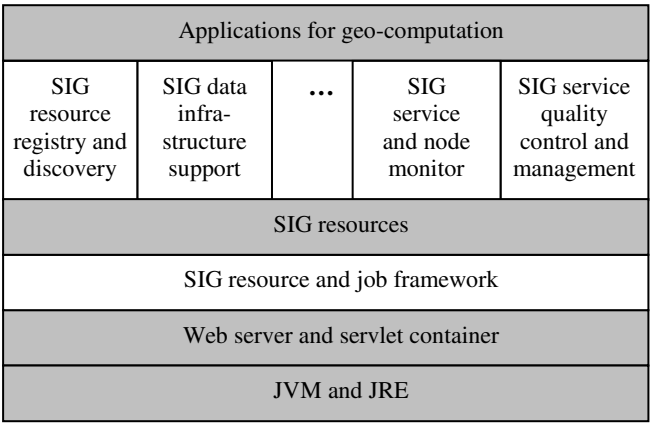


Fig. 1. Architecture of SIGRE

Besides the job framework, SIGRE provides some more components to make the runtime environment more useful and autonomic. They are: SIG resource registry and discovery service, SIG data infrastructure support, SIG service and node monitor framework, SIG service quality control and management, etc. The components provides a lot of basic functions for resources and applications, such as to collect and distribute the information about services ad nodes, monitor the execution of resources

and nodes, index and discover SIG resources, and control and manage service quality which can make SIG more friendly and autonomic.

3 The Autonomic Resource Distribution, Discovery and Matching

It is very important for users and applications to find and assemble the most suitable resources. It depends on several functions provided by SIGRE: resource discovery, service and node monitor, service quality control and management, etc. The figure 2 shows information flows for the autonomic resource distribution, discovery and matching.

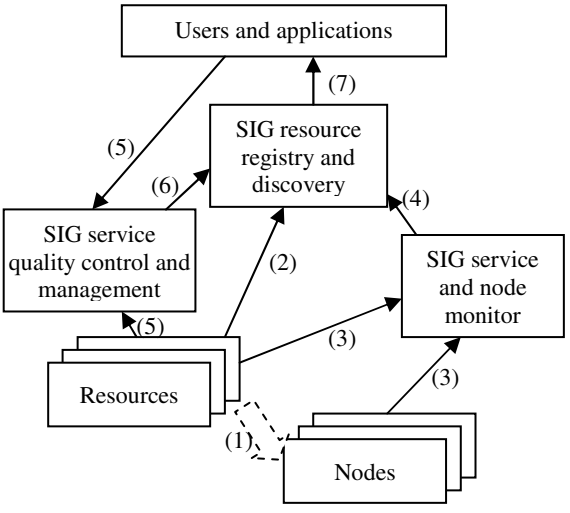


Fig. 2. Information flows for the autonomic resource distribution, discovery and matching

First of all, when the resource are deployed on nodes, (1) the static information about the resources such as its name, usage and function will be registered to the resource registry and discovery service. (2) Then, when the resources and services are invoked executed, the performance and status information about them will be probed and collected by the SIG service and node monitor. (3) And the information organized and analyzed by the SIG service and node monitor will be provided to the SIG resource registry and discovery service for service discovery and matching. (4) Furthermore, the quality and cost of SIG service may be different far away, users may care about them. So, the SIG service quality control and management will collect the service quality information from resources, users and applications, analyze them to get the score, (5) and provide the score to the SIG resource registry and discovery component so that the users and applications can find the most suitable and high quality resources by the resource matching. (6) Finally, integrating the static and dynamic information from SIG resources, service quality control and management component, service and node monitor, and other parts of SIGRE, the SIG resource registry and discovery service can discover and match the most suitable SIG resources for users and applications request. (7)

Around the SIG resource registry and discovery service which is extended based on the Distributed Grid Resource Registry Meta-Service [4] for geo-computation users, SIG service and node monitor, SIG service quality control and management component, and all the deployed SIG resources make up a closed loop. By the closed loop, the static, history and real-time information about all SIG resources, node status, service status, service quality, and so on, is collected, analyzed, and proceeded for the

resource matching. Then, the users and applications can find the resources needed: the resource with the highest performance, the best quality, the lowest cost, the most robust node, or the widest network bandwidth. With the distributed architecture of SIG, the closed loop enables the SIGRE to self-configure, self-manage, self-heal, and self-optimize in the resource discovery and matching.

For example, an application can find a data processing resource based on a super computer which has at least 16 processors and 8GB memory, a data source which provides Landset7 image through network with bandwidth at least 10Mbps, and a WMS service to visualize the processing result. SIGRE and the resource discovery service will search in the SIG resources by a well-defined p2p protocol and find the available and most suitable resources for the application. What the application need do only is to access the resources found and resolve problems for the end users. SIGRE provides an autonomic run time environment for the SIG applications.

In the other hand, the development and distribution of resources is very simple, too. Most of the SIG computation resources are implemented as SIG jobs which is deployed in the SIG job framework and managed by the SIGRE. As soon as a java class or a script file which implement the job and a job description file are deployed in the job framework, a SIG job is deployed and can serve the users and applications. And all information about the job will be collected, analyzed and used by the registry and discovery service, the service quality control and management component and other parts of SIGRE. SIGRE can configure and manage itself and resources based on it. It provides an autonomic runtime environment in resource distribution, discovery and matching for SIG resources and applications.

4 The Autonomic Data Infrastructure Based on SIGRE

Data sharing and distribution is another important usage for Spatial Information Grid. And it plays the role of bridge between the data consumer and data supplier, and brings spatial data on the finger of users. It is required for almost all the SIG applications to search and access spatial data through a powerful and easy-to-use data infrastructure. For the construction of SIG data infrastructure, a data sharing framework for the support of data infrastructure is included in the SIGRE. Based on it, a data distribution system is designed and implemented. [5] The data distribution system is built up by data sources, data agencies, registry services in SIGRE, client support libraries, and other components. Applications can query and access stored data by a well-defined XML based extensible RSI-data source accessing language through SOAP protocol which is platform-independent.

In the data infrastructure, each data provider is organized as a data source which has a uniform service accessing point. The service accessing point is a web service which is described by a WSDL [6] document and invoked through SOAP [7] protocol. By invoking the web service in a well-defined XML based protocol; data stored in the data node can be searched and accessed.

Although users and applications can search and download data from the data sources directly, it is too deficient for the SIG based data infrastructure which is built up by data nodes only to be an ideal data infrastructure for geo-computation. The third layer – agency layer is the most important layer to make the data infrastructure

flexible, extensible, autonomic, and powerful. In this layer, there are a series of data agencies with different goal and different functions serving for the users of the data infrastructure. For instance, a "catalogue agency" may collects information from data sources, generates a "catalogue" of data in the infrastructure, and provides a service for users to find data with some given features in the "catalogue". It is a "search engine" like "google" in the data infrastructure, and provides more powerful functions for the data infrastructure.

In order to make the design of data infrastructure simple and neat, the data agencies are required to adopt the same protocol as the data sources. It is called "eXtensible Data Accessing Language", shortly XDAL. Users can accomplish the operation by invoking the web service provided by data source or data agency, passing the request in XDAL format to it, and analyzing the response in XDAL format for the result. Figure 3 is a sample of searching data from satellite "Landset-7" with a given acquirement date.

Comparing with the ordinary data distribution system, the SIG based data infrastructure can distribute data with different types, different sources, and different goals in a uniform interface easily and flexibly. Furthermore, based on the data infrastructure support of SIGRE, a lot of "data agencies" will be continuously deployed in the data infrastructure and make the infrastructure more flexible, extensible, autonomic, and powerful.

1、client sends the request of searching an image

```
<query>
  <conditions relation="AND">
    <condition op="EQ">
       <param>satellite</param>
      <value>landset7</value>
    </condition>
    <condition op="EQ">
       <param>date</param>
       <value>2006-01-01</value>
    </condition>
  </conditions>
  <orders>
    <sortBy order="ASC">ID</sortBy>
  </orders>
</query>
```

2、data source starts the operation, and returns a operation ID

```
<response>
   <operationID>001235864631</operationID>
</response>
```

3、client gets the operation status

```
<getStatus>
  <operationID>001235864631</operationID>
</getStatus>
```

4、data source returns the operation status

```
<status>
 <operationID>001235864631</operationID>
 <currentStatus>processing</currentStatus>
</status> <!-- processing -->
<status>
  <operationID>001235864631</operationID>
  <currentStatus>finished</currentStatus>
</status> <!-- finished -->
```

5、client gets the operation result

```
<getResult>
  <operationID>001235864631</operationID>
</getResult>
```

6. data source returns the operation result

```
<result>
  <operationID>001235864631</operationID>
  <resultSet>
    <item>
      <id>0013256782</id>
      <spacecraft>Landset7</spacecraft>
      <senser>ETM+</senser>
       ……
    </item>
      ……
    <item>
      <id>0020165784</id>
       ……
    </item>
  </resultSet>
</result>
```

Fig. 3. Sample of searching data

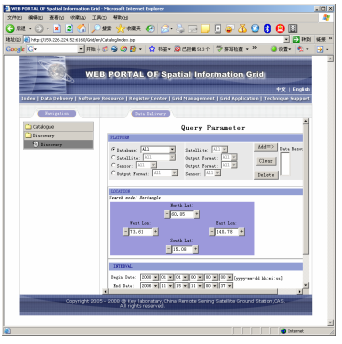
5 Implementation, Test and Future Works

Java language is often adopted by grid implementation because of its platform-independent, acceptable performance and mass of open-source support. So, the SIGRE is implemented based on the famous SOAP support library Axis [8] by java language, too. For the sake of easy deployment, the SIGRE implementation is distributed as a WAR package. It can be deployed on most of the popular servlet container such as Apache Tomcat [9] and

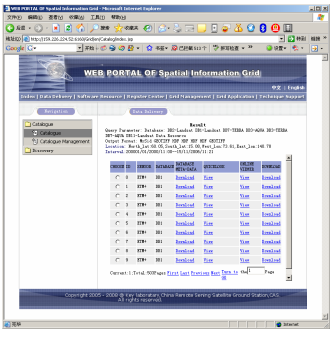
provide an autonomic runtime environment for not only SIG applications but also SIG resources. Based on the SIGRE, a SIG testbed is constructed and deployed on Apache Tomcat servlet container. The testbed integrates three kinds of computing nodes (high performance cluster, condor computing pool, and traditional high-end server, totally more than 100 processors) for its computing infrastructure, about ten data sources with more than 10TB data for its data infrastructure, more than forty algorithms for data processing models, several WMS servers for data visualization, a web based management and monitor user interface for management, and so on. Figure 4 shows some of the SIG testbed user interfaces.

The test on SIG testbed shows that SIGRE can provide a powerful, easy-to-use, robust and autonomic runtime environment for SIG, developers can develop SIG resources and SIG applications on SIGRE easily and quickly. For example, based on the SIGRE data infrastructure support component, a new data source can be developed and deployed successfully in one day at most, and to create a new application based on SIGRE may take only hours or less.

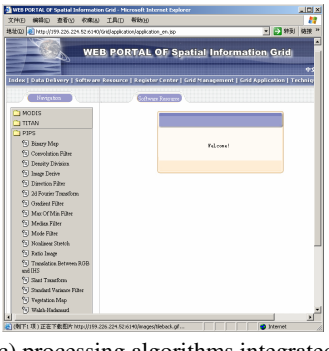
As a kind of application grid for geo-computation context, Spatial Information Grid needs a powerful and easy-to-use runtime environment for its users and developers. In this paper, based on the techniques including web service, grid computing, and spatial information grid, an autonomic runtime environment named SIGRE is proposed and implemented for geo-computation. Each SIG node carries out an instance of SIGRE, which provides the basic running environment including the java virtual machine, runtime libraries, management and optimization components, monitoring and healing components, etc. All SIGRE instances on the SIG nodes makeup an environment for the execution and management of resources and applications in SIG, and provide the key functions including resource discovering, data managing, service quality supporting, etc. based on SIGRE, a SIG testbed is constructed and tested. The test shows that SIGRE can provide a powerful, easy-to-use, robust and autonomic runtime environment for SIG, and developers can develop SIG resources and SIG applications on SIGRE easily and quickly.



(a) input data searching arguments



(b) data searching result



(c) processing algorithms integrated

Fig. 4. User interfaces of SIG testbed

But, the SIGRE can not make the SIG popular immediately and automatically. A lot of useful and powerful applications are needed to respond the requirement of SIG end users. It is very important for the survival and growth of SIG to provide mass of applications for the users to resolve their problem.

The requirements of the end users are so various that it is difficult to respond them by limited developed applications. The application development becomes the bottleneck of SIG popularization. One way to ease the bottleneck is to develop and deploy more and more powerful applications as soon as possible and as many as possible so that the requirements of end users can be responded. It is the traditional way and seems failed. There are two main reasons at least. First, if the application is more powerful, it is more complex, more expensive, and more difficult to develop. Second, the requirement of end user is always uncertain and unstable. It is too difficult to develop a powerful application for the uncertain and unstable requirement of a mass of end users with a low cost.

Another possible way to ease the bottleneck is to make the application development easier so that more people, even all end users can develop applications for themselves. Situational application will be a good choice. In SIG, situational application is such an application that it is quickly created and deployed based on the SIG infrastructure and SIG resources for a situational requirement by the end user. It is often developed and deployed by a series of visual development toolkits. To make the SIGRE support the development and deployment of situational applications will be one of our most important future works to make the SIG more useful and popular. Based on the SIGRE situational application support, users can create and deploy situational applications for their requirements themselves. It will ease the application development bottleneck of SIG and make SIG more powerful and popular.

References

1. Java Technology, http://java.sun.com
2. Java Servlet - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Java_Servlet
3. Huang, Z.C., Li, G.: An SOA based On-Demand Computation Framework for Spatial Information Processing. In: GCCW 2006. Fifth International Conference on Grid and Cooperative Computing Workshops, Hunan, China, October 21-23, 2006, pp. 487–490 (2006)
4. Huang, Z.C., Du, B., Gu, L., He, C., Li, S.: Distributed Grid Resource Registry Meta-Service: Design and Implementation. In: The 7th International Symposium on Autonomous Decentralized Systems, Chengdu, China, April 4-6, 2005, pp. 531–535 (2005)
5. Huang, Z.C., Li, G.: SIG-DDS: A Grid-based Remote Sensing Data Distribution System. In: SKG 2006. Second International Conference on Semantics, Knowledge, and Grid, GuiLin, China, November 1-3, 2006, pp. 93–94 (2006)
6. Web Services Description Language (WSDL) 1.1: http://www.w3.org/TR/wsdl
7. SOAP: The fundamental message enveloping mechanism in Web services, http://www.w3.org/TR/SOAP
8. The Apache Software Foundation: Web Services - Axis, http://ws.apache.org/axis/
9. The Apache Software Foundation: Apache Tomcat, http://tomcat.apache.org/

A Flexible Job Scheduling System for Heterogeneous Grids*

Lan Cheng, Hai Jin, Li Qi, and Yongcai Tao

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

Abstract. Job management is the most complicated and kernel component of grid system. However, due to the dynamic, heterogeneous and dispersed nature of grid environment, grid job submission and scheduling are always intractable issues that needs to be addressed, especially across heterogeneous grids. In this paper, a *Flexible Job Scheduling System* (FISS) is designed, which exploits and extends JSDL to support the job submission with different QoS requirements across heterogeneous grid platforms. Based on the extended JSDL, multiple types of jobs are supported, including WS, WSRF, GRS. Moreover, it conduces to the interoperation among heterogeneous grid systems. Experiments are carried out and proven that FISS proposed in the paper can improve the efficiency and utilization of grid system while satisfying users' QoS requirements and achieving the interoperability among heterogeneous grids.

1 Introduction

Grid computing is emerging as a novel infrastructure for the coordinated resource sharing, problem-solving and services orchestration in dynamic, multi-institutional *Virtual Organization*s (VOs) by integrating large-scale, distributed and heterogeneous resources [1, 2]. Users, ranging from scientific communities, business communities to general consumers, are utilizing grids to share, manage and process large data sets and construct large-scale applications. Job scheduling is the most complicated and the kernel component of a grid system. It accepts users' job requests, and then interacts with information center to select and invoke relevant services according to specific QoS requirements and scheduling policy. Due to the dynamic, heterogeneous and distributed nature of grid environment, grid job scheduling is confronted with significant challenges such as security, quality of service, and lack of central control within distributed virtual organizations.

In addition, with the rapid development of the grid, a wide variety of grid middlewares and grid systems have been developed in numerous research projects all over the world, such as Globus Toolkits [3], CGSP [4], UNICORE [5], GOS [6]. Though

* This paper is supported by National Science Foundation of China under grant 90412010 and China CNGI project under grant CNGI-04-15-7A.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 330–339, 2007.

developed based on the idea of OGSA which is the de facto standard of grid, most grid systems have its own implementations for specific application. Hence, existing grid systems differ from each other in management so that the interoperability among them is poor. Recently, interoperability among heterogeneous grids is a hot research point attracting more attentions, which can integrate more grid resources and eliminate the grid resource island. However, to achieve interoperability among heterogeneous grids, some issues must be addressed such as security, information service, job management, and data management. Because most of grid platforms are built for specific applications, the types of jobs supported by each grid vary greatly. Even though the jobs in different grid platforms provide similar functionality, the access interface and types of parameters are different greatly, which bring new challenges to grid job scheduling.

A grid job scheduling system must be able to provide uniform job submission interface and support various types of jobs, shielding the heterogeneity of grids and making grid systems look the same. There are already different languages to describe and submit grid jobs, e.g. JDL of EGEE [7], Globus RSL [8]. However, these job description languages are designed for particular projects and can not interoperate each other.

In this paper, a *Flexible Job Scheduling System* (FISS) is designed, which exploits and extends JSDL to support job submission of various types of jobs, such as WS, WSRF, and GRS. Moreover, it conduces to the interoperability among heterogeneous grid systems by adopting virtualization layer and plug-in technologies. Experiments are carried out and experimental results prove that FISS proposed in the paper can improve the efficiency and the utilization of a grid system while satisfying users' QoS requirements and achieving the interoperability among heterogeneous grids.

The rest of the paper is organized as follows. Section 2 reviews the related work. We propose FISS in section 3. Section 4 introduces the components of FISS. The experimental evaluation is presented in section 5. Finally, we conclude and give some future work about our research in section 6.

2 Related Work

There are already different languages to describe grid jobs. Some of them are included in large projects and adapt to the project requirements, such as *Job Description Language* (JDL) for *Enabling Grids for E-sciencE* (EGEE), *Globus Resource Specification Language* (RSL). The European Data Grid JDL is proposed in the context of the European Data Grid Project and afterwards adopted by the EGEE project [10]. It is based on the classed language and can be used as the language substrate of distributed frameworks. JDL allows specifying grid job attributes such as Job Type, Executable, Arguments, Stdinput/Stdoutput. The *Globus Resource Specification Language* (RSL) provides a common interchange language to describe resources and jobs. The current version is RSL-2, namely, the Web Services versions (GT3 and GT4). RSL-2 is based on XML technology and allows specifying a more extended set of attributes than its predecessor.

The above mentioned job description languages are designed for their own applications so that they are poor at interoperation. Since there are a lot of different

languages for describing grid jobs, OGF has presented the *Job Submission Description Language* (JSDL) to standardize the job submission language [11]. We believe that the JSDL is a good solution but it has some deficiencies regarding the interoperability issues among heterogeneous grids.

Due to the diverse failures and error conditions in grid environments, scheduling in grid environment is an NP-complete problem, and many heuristics algorithms have been proposed to obtain the optimal scheduling, such as Min-min, Max-min [9]. Existing grid job scheduling algorithms have some features in common, consisting of two main phases. In the first phase, while receiving user's job request, job manager would interact with information center and the set of qualified service resources are selected. In the second phase, job manager would choose the optimal resource from the set of qualified resources according to specific scheduling algorithm while considering both user's QoS requirements and system QoS features.

In this paper, JSDL is exploited and extended to support uniform job submission across heterogeneous grid platforms. In addition, a novel job scheduling algorithm is presented to optimize the job scheduling among different grid platforms.

3 FISS Architecture

In this paper, a *Flexible Job Scheduling System* (FISS) for heterogeneous grid interoperability is presented. FISS creates a virtual job management center to implement job submission and scheduling among heterogeneous grid platforms. Figure 1 shows the FISS architecture. It mainly comprises four components: JSDL parser, job waiting queue, scheduler and plug-ins. In order to make FISS compatible with other grid system, JSDL is adopted and extended as the job submission description language. JSDL parser is used to parse the job's JSDL document that users submitted. Furthermore, it queues users' job by adopting batch scheduling mode. Scheduler schedules the jobs balancing trade-off between users' QoS requirements and resource utilization. The plug-ins is mainly used to bridge the virtual management center and other heterogeneous grids. The function of plug-ins includes converting the parameter types, integrating the service information from various heterogeneous grids, mapping user identity and data transferring.

4 FISS Implementation Techniques

In this section, the key components of FISS will be introduced and discussed.

4.1 JSDL Parser

Since there are various job submission languages for a variety of job management systems, the main goal of JSDL is to standardize the job submission. It is used to describe the requirements of computing-intensive jobs for submission to resources, particularly in grid environments. JSDL language comprises a vocabulary and normative XML schema that facilitate the expression of those requirements as a set of XML elements.

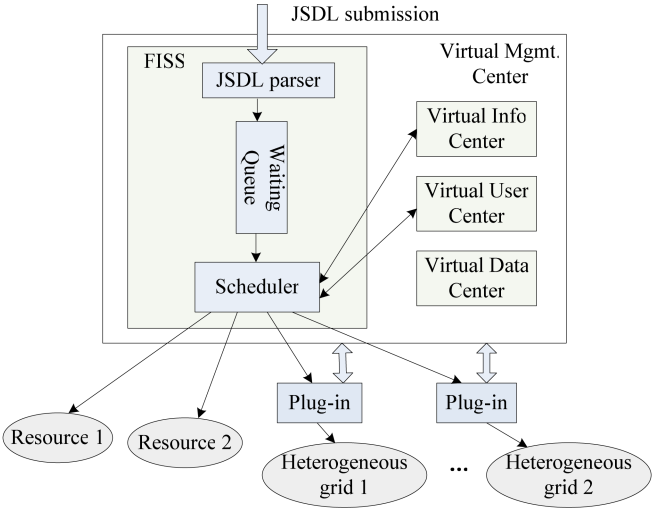


Fig. 1. FISS Architecture

The JSDL 1.0 elements fall into the following general categories: job identification requirements, resource requirements, data requirements and extension.

```
<JobDescription>
        <JobDescription>
                <JobIdentification … />?
                <Application … />?
                <Resources … />?
                <DataStaging … />*
        </JobDescription>
        <xsd: any # # other>*
</JobDescription>
```

The *JobIdentification* element contains all elements that identify the job: *JobName, Description*, *JobAnnotation*, and *JobProject*. If this element is not presented then its value, including all of its sub-elements, is undefined. The *Application* element describes the application and its requirements. It contains the name of the application, the version and a description. The *Application* element includes only the generic elements and more specific application definitions should be defined through specific extensions (i.e. POSIX compliant normative extension). The *Resources* elements describe the resource requirements of the job. The *DataStaging* element defines the files that should be moved to the execution host (stage in) and the files that should be moved from the execution host (stage out). Files are staged in before the job starts executing and staged out after the job terminates. JSDL provides the overall structure

to define the submission requirements of jobs. This structure may be extended to best fit more specialized needs.

JSDL works efficiently and flexibly for job submission in homogeneous grid system, but poor in submitting across heterogeneous grid platforms. To solve this problem, we extend JSDL in FISS. JSDL provides two mechanisms for extension: using attributes and using elements. In the paper, we extend JSDL based on adding new elements.

```
<jsdl:PlatformSelection number="***">
<jsdl:Platform>
          <jsdl:Value> ***</jsdl:Value>
          <jsdl:URI> *** </jsdl:URI>
</jsdl:Platform>
<jsdl:Platform>
          ……
</jsdl:Platform>
</jsdl:PlatformSelection>
```

In the extension part of JSDL, the attribute *number* specifies the number of heterogeneous grid platforms onto which jobs are submitted. The element *Platform* describes the relative information of specified grid platform, including the platform's name, resource access URI, and so forth.

In order to complement JSDL to flexibly support various types of jobs, we add new element in JSDL as follows.

```
<jsdl:SelectedServiceTypes>
<jsdl:Value> WS</jsdl:Value>
<jsdl:Value> WSRF</jsdl:Value>
<jsdl:Value> GRS</jsdl:Value>
</jsdl:Platform>
```

With JSDL, a user can specify the job information and QoS requirements. For the extended JSDL, we design a JSDL parser, which parses the job JSDL description and queues the job in different job waiting queues according to the QoS requirements.

4.2 Job Waiting Queue

In order to efficiently schedule jobs to resources belonging to different grid systems with consideration of users' QoS requirements and resource utilization, FISS exploits batch scheduling mode, namely, only while the amount of jobs reach certain number or after a certain time interval, is the scheduler triggered. Batch scheduling mode improves system utilization while satisfying users' QoS requirements.

4.3 Scheduler

Scheduler is one of the key components of FISS, and it is in charge of finding the mapping between jobs and grid resources. Scheduler fetches the job from the job waiting queue. Then, by adopting novel scheduling algorithm similar to Max-min and Min-min algorithms, FISS interacts with virtual information center and selects the optimal resources of grid platforms and makes scheduling decision. The rational is that the completion time of jobs on each resource is first predicted, then like Min-min algorithm, the job having the minimum *estimated completion time* (ECT) value is chosen to be scheduled onto the resource on which job's EST is minimal. Afterward, similar to Max-min algorithm, the job having the maximum EST value is chosen to be scheduled onto the resource on which job's EST is maximal.

To predict the completion time of jobs, we adopt the following model:

$$\mathrm{EST_{ij}} = \mathrm{WT_{ij}} + \mathrm{ET_{ij}} + \mathrm{DT_{ij}} \tag{1}$$

The model composes of three parts. WT_{ij} denotes the waiting time of job i at resource j, namely, it must wait until all jobs in waiting queue are finished. ET_{ij} represents the execution time of job. DT_{ij} denotes the data processing time in order to executing job, including the time of fetching the input data and the one of outputting result data.

According to above rule, jobs are scheduled. The scheduling algorithm is shown in Algorithm 1. The algorithm consists of three parts. First, the minimal ECT of jobs is found. Then, Min-min and Max-min algorithms are alternately utilized to schedule jobs. Meanwhile, according to job's JSDL information, if the user specifies the destination scheduling platforms, job will be scheduled onto the resource of these grid platforms respectively. On the contrary, if the user does not specify, job will be assigned to the optimal resource regardless of grid platforms.

Algorithm 1. FISS job scheduling algorithm

```
For each job i in waiting queue do
For each resource j do
          ECT_ij = CT (job i, resource j);
          MinECT_ij = Min (MinECT_ij, ECT_ij);
    Endfor
Endfor
flag = true;
While ∃job not scheduled in waiting queue do
    If (flag) {
For each job i do
               MinMinECT_ij = Min (MinMinECT_ij, MinECT_ij);
         Endfor
         For k=1 to JobJSDL.PlatformSelection.number do
```

```
                Platform[k].Resource j ←Job i; // ECT of
job i in resource j is minimum
          Endfor
           Resource j ← Job i;
flag = ! flag;
      }
      If (flag) {
           For each job i do
                MaxMinECT_ij = Max (MaxMinECT_ij, ECT_ij);
          Endfor
          For k=1 to JobJSDL.PlatformSelection.number do
                Platform[k].Resource j ← Job i; // ECT of
job i in resource j is minimum
          Endfor
          Resource j ← Job i;
          flag = ! flag;
      }
Endwhile
```

4.4 Plug-in

Plug-in plays an important role in FISS, which serves as a bridge between virtual grid management center and heterogeneous grid platforms. The main function of plug-in is to retrieve the information data from the information centers of heterogeneous grids and translate the data to consistent data representation stored in virtual information center. In addition, plug-in is responsible for converting the service accessing interface and types of parameter and data transfer between heterogeneous grids. The introduction of plug-in improves the scalability and extensibility of system.

5 Performance Evaluation

5.1 Experimental Environment Settings

We evaluate the validity and performance of FISS in a real heterogeneous grid environment. The testbed includes six nodes, four are in *Cluster and Grid Computing Lab* (CGCL), and two are in *National Hydro Electric Energy Simulation Laboratory* (NHEESL). Grid nodes are respectively deployed by two heterogeneous grid systems: CGSP [4] and VEGA [6], which are two different grid middleware in China. Their configurations are shown in Table 1. We compare FISS with two common scheduling algorithms First-fit and Min-min as follows:

- First-fit: It belongs to first-come first-service strategy, and schedules the arrived job to the optimal resource according to the order of job arriving.
- Min-min: In Min-min mode, the job with the minimum *estimated execution time* (EST) value is selected. Then it is scheduled to the resource on which the EST value is minimal [12].

Table 1. Experimental Environment

Nodes / Metric	Node 1	Node2	Node3	Node4	Node5	Node6
CPU	P3 1GHz	P3 1GHz	P3 2GHz	P3 2GHz	IA 64 1.3GHz	IA64 1.3GHz
Memory	512MB	512MB	512MB	2GB	2GB	2GB
Grid system	CGSP	CGSP	CGSP	VEGA	VEGA	VEGA
Location	CGCL	CGCL	CGCL	CGCL	NHEESL	NHEESL

5.2 Experimental Results

We compare FISS with First-fit and Min-min while scheduling jobs across different grid platforms in terms of success ratio of jobs and system throughput. We test three types of applications in our grid environments, which are both computing-intensive and data-intensive. One is gene sequence matching application (App.1). The second is image processing application (App.2). The third is video conversion (App.3). In order to validate the extended JSDL, we deploy App.1 with Web Service, App.2 with WSRF and App.3 with GRS. 200 different types of jobs are respectively run at different time interval.

Figure 2 shows the success ratio of three types of grid applications. We can conclude that FISS has the highest success ratio and First-fit has the lowest. It is because that FISS considers the trade-off between large jobs and small jobs, avoiding the long waiting of large jobs. First-fit does not belong to batch scheduling mode and can not consider the QoS characteristics of both jobs and grid resources. Min-min may result to that long jobs waiting too long time and finally fail.

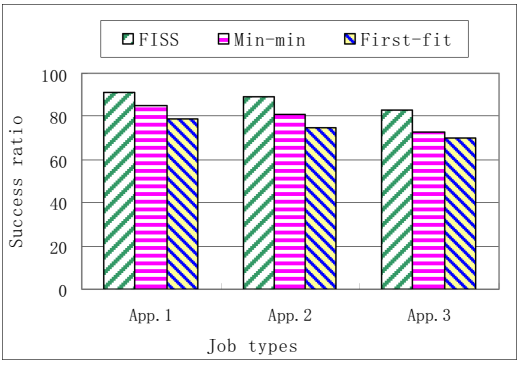


Fig. 2. Success Ratio of Grid Applications

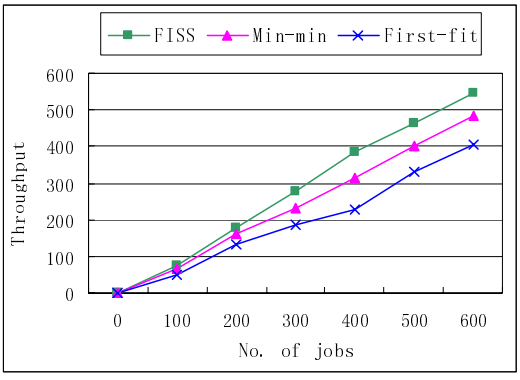


Fig. 3. System Throughput

Figure 3 shows the system throughput of three different scheduling algorithms. It is clear that with the number of submitted jobs increases, FISS is superior to both Min-min and First-fit. This is because that FISS can efficiently utilize grid resources with the increasing of submitted jobs.

6 Conclusions and Future Work

Grid computing is different from conventional distributed systems by integrating large-scale resources which are mostly heterogeneous. Existing grid systems are mostly developed for particular application and are poor in interoperation, which results in new grid resource islands. In this paper, a *Flexible Job Scheduling System* (FISS) is designed, which can provide efficient job submission and scheduling across heterogeneous grids. In view of compatibility with other grids and enabling integration of more grid platforms in the future, FISS exploits and extends current widely-accepted standard, *Job Submission Description Language* (JSDL), for job description. Considering the trade-off between users' QoS requirement and system utilization, FISS adopts novel scheduling algorithm. Experimental results prove that FISS can achieve the interoperation among heterogeneous grids and meanwhile improve the system utilization while satisfying users' QoS requirement. As our future work, we plan to perfect FISS and to validate its efficiency further.

References

1. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure, 2nd edn. Morgan Kaufmann, November (2003)
2. Hwang, S., Kesselman, C.: Grid Workflow: A Flexible Failure Handling Framework for the Grid. In: HPDC 2003. Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing, Seattle, Washington, USA,, June 22-24, 2003, IEEE Computer Society Press, Los Alamitos (2003)

3. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
4. ChinaGrid Support Platform, http://www.chinagrid.edu.cn/cgsp/
5. Erwin, D. (ed.): UNICORE Plus Final Report - Uniform Interface to Computing Resources. The UNICORE Forum (2003) http://www.unicore.org/documents/UNICOREPlus-Final-Report.pdf
6. Xu, Z., Li, W., Li, Z., Yu, H., Liu, D.: Vega grid: A computer systems approach to grid research. In: Proceedings of the 2nd International Workshop on Grid and Cooperative Computing, Shanghai, pp. 480–486 (2003)
7. Pacini, F.: Job Description Language How-to, http://server11.infn.it/workload-grid/ docs/ DataGrid-01-TEN-0102-0_2-Document.pdf
8. Globus Resource Specification Language RSL v.1.0. http:// www-fp.globus.org/ gram/ rsl_spec1.html
9. Mandal, A., Kennedy, K., Koelbel, C., Marin, G., Mellor-Crummey, J., Liu, B., Johnsson, L.: Scheduling strategies for mapping application workflows onto the grid. In: Proceedings of Fourteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-14), North Carolina, USA, July 24-27, 2005, pp. 125–134. IEEE Computer Society, Los Alamitos (2005)
10. Enabling Grids for E-sciencE (EGEE) Web Site, http://public.eu-egee.org/
11. GGF JSDL Working Group Web Site, https://forge.gridforum.org/projects/jsdl-wg/
12. He, X., Sun, X., von Laszewski, G.: QoS guided min-min heuristic for grid task scheduling. Journal of Computer Science and Technology 18(4), 442–451 (2003)

n-Cube Model for Cluster Computing and Its Evaluation

Tian Song, Dongsheng Wang, Meizhi Hu, and Yibo Xue

Tsinghua University, Beijing, 100084, P.R.China
{songt02,hmq02}@mails.tsinghua.edu.cn

Abstract. Cluster systems are widely used in modern high performance computing. With the rapidly increasing of parallel algorithms, it is an open problem to analyze and evaluate whether they take good advantage of the computing and network resources of clusters.[1−3] We present a novel mathematic model(n-Cube Model for Cluster Computing) that epitomizes the algorithms commonly used on clusters and evaluate this model using Stochastic Petri Nets (SPN). The state space of our model's SPN is also discussed formally. Finally, we take MM5(the Fifth-Generation Model) as a case and the comparative performance analysis shows the immense vitality of the model.

1 Introduction

Clusters in a local network with high-speed interconnections are more and more popular in modern high performance computing. They have such properties as obtaining high performance at a low price and good scalability. It is available and economical to solve large scientific and engineering problems on clusters.

With many applications move to clusters, it is an open problem to analyze and evaluate whether the parallel algorithms take good advantage of the computing and networking resources of clusters.[1−3] The performances of most algorithms are traditionally evaluated through simulation instead of theoretic analysis since they emphasize on implementation.[1][3] With the actual case that many algorithms on clusters use the same essence of topology, some models suitable to theoretic analysis and evaluation have been presented nowadays.[2]

In this paper, we present a novel mathematic model that epitomizes the algorithms commonly used on clusters, in which each peer exchanges mass data with and only with all its logic neighbors and deals with computing respectively. We construct a mathematical model based on n-Cube whose structure is good for theoretic analysis and evaluation. We name this model n-Cube Model for Cluster Computing(nCMCC). The main feature of this model is to gain the maximal parallelism by regulating the communication. And the model has the same topology essence with most cluster algorithms. Therefore, the analysis and evaluation of its performance correspondingly make sense to the actual applications.

The rest of the paper is organized as follows. In section 2, the definition of our n-Cube Model for Cluster Computing(nCMCC) is presented and some properties

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 340–351, 2007.

of it are also discussed. In section 3, we give the method to construct Stochastic Petri Nets from the nCMCC in theory. And in section 4, we take MM5(the Fifth-Generation Model)[10−12] as an application example and evaluate its performance using Stochastic Petri Nets. Finally, conclusions are given in section 5.

2 n-Cube Model for Cluster Computing

n-cube is a hypercube with n dimensions and 2^n nodes. Every node in the n-cube is exactly connected with another n nodes. This kind of structure can be used to describe a kind of algorithms on clusters, in which each node exchanges mass data with the same number of logic neighbors and deals with the data. The algorithms usually include communication and computing. Since computing is done respectively by nodes, we mainly emphasize on the modeling of the communication process. We name this kind of model n-Cube Model for Cluster Computing (nCMCC), since it is based on n-cube structure.

Definition 1. *The nodes of the cluster is represented by the point of the n-cube and the communication path is represented by the edge of the n-cube. We define this model as a set of* $\{ Va, Vb, E, R \}$ *where*

- $Va \bigcup Vb$ *is the set of n-cube nodes,* $Va \bigcap Vb = \phi$*, the* ***node rule*** *gives the details on how to sort all the nodes into Va and Vb;*
- $E = (Va \times Vb) \bigcup (Vb \times Va)$*, its members are the edges of the n-cube, which represent the possible communication paths of the nodes in clusters.*
- *R is the set of communication rules of the nodes in clusters. Detail information see the* ***communication rule****.*

Node rule: This rule gives details on how to sort all nodes into Va and Vb.

- Establish a n-dimension coordinate space using the edges of n-cube and then code all the nodes with the format $(b_1 b_2 \cdots b_n)$, where b_i is a binary number.

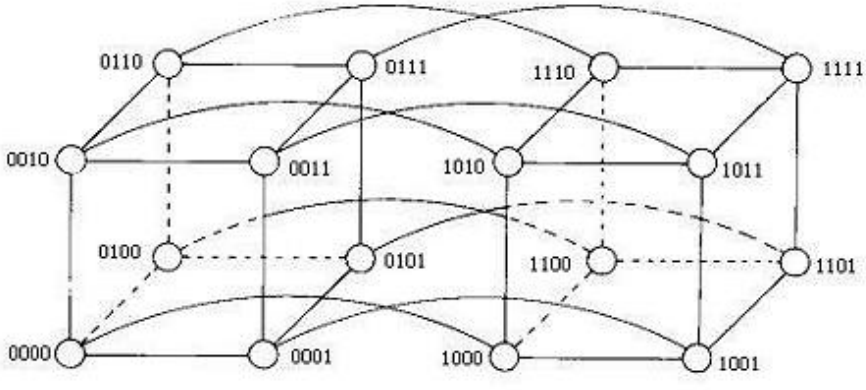


Fig. 1. Example of n-Cube Model for Cluster Computing, where n=4

– Va = $\{(b_1b_2\cdots b_n) \mid (b_1 \oplus b_2 \oplus \cdots \oplus b_n) = 0\}$, Vb = $\{(b_1b_2\cdots b_n) \mid (b_1 \oplus b_2 \oplus \cdots \oplus b_n) = 1\}$. That is, $\forall$node$(b_1b_2\cdots b_n)$, if there are even number of 0 in$\{b_1,b_2,\cdots,b_n\}$, then the node belongs to Va; on the contrary,if there are odd number of 0, it belongs to Vb. ($\oplus$ is a XOR operation.)

Communication rule: This rule gives a mathematical description of the communication between the nodes on clusters.

– The communication happens only between the nodes of Va and the nodes of Vb. The nodes of the same set can't communicate with each other.
– At the beginning, Va receive data from Vb. After this stage is over, Va change to send data and Vb begin to receive. While all these data exchanging is finished, the communication phrase is over and the nodes begin to compute the answers respectively.
– No matter receiving and sending data, the orderliness of the communication is the same. As for node $(b_1b_2\cdots b_n) \in Va$, i$\in\{1,2,\cdots,n\}$, the i^{th} communication is between this node and the node $(b_1b_2\cdots \bar{b_i}\cdots b_n)$. ($\bar{b_i}$ is the complementary of b_i)

The n-Cube, whose nodes are labelled by binary numbers, is called **Boolean n-Cube**, which is a tool for the theoretic research of interconnection networks for multiprocessor systems.[13−15] The theory of Boolean n-Cube can further expend the nCMCC in many aspects[13] . Figure 1 gives a 4-Cube Cluster Algorithm Model for example.

Theorem 1. *Suppose the communication traffic to each node on clusters is the same, the nCMCC can gain the maximal parallelism during the communication.*

Proof. Since the communication exists between two nodes and the communication traffic to each node is the same, the most parallelism means that all the nodes communicate with one another without waiting. As for nCMCC, if the communication process is a one-to-one(conformal) mapping between Va and Vb at one step, we can proof the maximal parallelism.

According to the *communication rule* of **definition 1**, the i^{th} communication is between $(b_1b_2\cdots b_i\cdots b_n)$ and the node $(b_1b_2\cdots \bar{b_i}\cdots b_n)$. For $\forall(b_1b_2\cdots b_i\cdots b_n) \in$ Va, there is only one corresponding node$(b_1b_2\cdots \bar{b_i}\cdots b_n)$ in Vb. For any $(b_1b_2\cdots \bar{b_i}\cdots b_n) \in$ Vb, there is also only one corresponding node $(b_1b_2\cdots b_i\cdots b_n)$ in Va. Thus the communication rule is a one-to-one mapping between Va and Vb. Therefore the nCMCC can gain the maximal parallelism while the nodes communicate with each other.

The nCMCC has many better properties for analysis and evaluation than the algorithms only for implementation. First of all, nCMCC can be accurately expressed. Binary encoding is exactly appropriate to represent the nodes in the n-cube structure since there are only two scales in every dimension. As long as the directions are determined, only one coding is qualified to every node. According to the *communication rule* described previously, it is very simple for one node to determinately know which one and how to be communicated next time. Thus the model is practical for both analysis and implementation.

Secondly the nCMCC has a very considerable feasibility and scalability. When the number of logic nodes increases, that is, when n increases, it is expected that nearly nothing needs to modify in the model.

The nCMCC can gain the maximal parallelism when the communication traffic to each node on clusters is the same. In nCMCC, the communications of 2^{n-1} pairs can be dealt with synchronously. This is the best case all parallel algorithms go in for while dealing with the communication. That is to say, this model can be a theoretic representation of all the algorithms pursuing the maximal parallelism.

3 Evaluation Using Stochastic Petri Nets

Stochastic Petri Nets(SPN) is a useful mathematical tool to represent and analyze complex systems with interdependent components.[4−9] It has many good properties such as concurrency, nondeterminacy, asynchronization, capacity to depict and analyze distributed systems and so on.

A Stochastic Petri Net (SPN) model associates with each transaction an exponential *firing time*.[8] This feature allows SPN an isomorphism with Markov chain and distinguishes from a normal Petri Net. In this paper we suppose the reader has some knowledge with SPN.

3.1 SPN Construction

nCMCC is a novel mathematic model for parallel computing. Based on n-cube, this model has a good structure for expression, definition and theory research. However n-cube model has some limitations to deal with time parameter and hard to represent dynamic transitions. Since SPN has many advantages in dealing with dynamic activities, it is chosen to be applied to analyze and evaluate our model.

Definition 2. *A Stochastic Petri Net can be constructed from nCMCC. With the exponential time parameters, nCMCC SPN* $= \{S, T; F, W, M_0, n, \lambda\}$*, where*

- $S = \{p_0, p_1, \ldots, p_\Omega\}$ *is the set of places,* $\Omega = (2n+1)2^n$*. Every node of the nCMCC corresponds (2n+1) places, in which 2n places are used for communication and 1 place for computing.* p_0 *is an additional place representing the end state of execution.*
- *T is the set of transitions, which represents the communication activities and computing activities.*
- $F = (S \times T) \bigcup (T \times S)$*, its members are described by arcs in the graphic.*
- $W{:}F \longrightarrow N^+$*, for* $f \in F$*,* $W(f) = 1$*, this is the weight function of arcs. N is the set of numbers.*
- $M_0{:}S \longrightarrow N$*, for* $\forall s \in S$*,* $M_0(s) = 0$*;*
- *n is the dimension of nCMCC.*
- $\lambda = \{\lambda_m, \lambda_c\}$ *is the set of expected firing rates.* λ_c *is the expected firing rate of computing transition and* λ_m *is the expected firing rate of communication transition.*

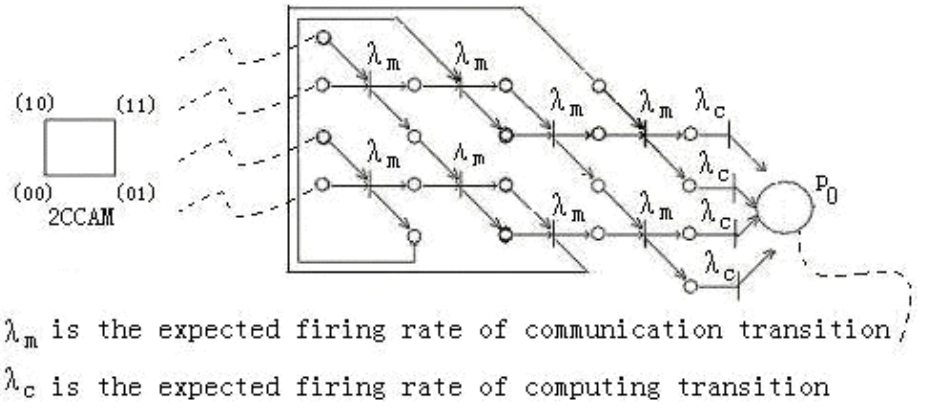


Fig. 2. Example of a 2-Cube Cluster Algorithm Model and its SPN

We take 2CMCC for example. 2CMCC is based on the 2-dimension cube, which has 4 nodes that can be coded as (00), (01), (10), (11). According to the *node rule* of **definition 1**, the 4 nodes are divided into Va = $\{a_1(00), a_2(11)\}$ and Vb = $\{b_1(01), b_2(10)\}$. Based on *communication rule* of **definition 1** and **definition 2** , we can construct the 2CMCC SPN. Figure 2 gives the detail of the graphic.

3.2 Some Utility Formulas

Stochastic Petri nets (SPN) are good tools for the performance evaluation. Since the nCMCC has been mathematically defined and the corresponding SPN can be easily constructed, we continue to analyze and evaluate the model's performance using SPN theory.

The MC of a given SPN can be automatic generated from the SPN.[5] We assume the number of states in the constructed MC is Ω (The value of Ω can be found in the **definition 2**.) and the state set of the MC is $\{M_i, \mathrm{i} = 1,2, , \Omega\}$. According to our nCMCC and the SPN theory, we can conclude the following formulas.

For nCMCC SPN = {S, T; F, W, M_0, n, λ}, $\forall\ p_i \in \mathrm{S}$, $\forall\ \mathrm{j} \in \mathrm{N}$, $\mathrm{P}\{\mathrm{M}(p_i) = \mathrm{j}\}$ represents the probability of j tokens in place p_i, we can conclude the function of probability density

$$P\{M(p_i) = j\} = \sum_k P\{M_k\} \tag{1}$$

Where, $\mathrm{P}\{M_k\}$ is the probability of steady state for M_k in Markov chain.

For $\forall\ p_i \in \mathrm{S}$, $\bar{u_i}$ denotes the expected number of tokens in a reachable place p_i, when the state is steady. Thus

$$\bar{u_i} = \sum_j j \times P\{M(p_i) = j\} \tag{2}$$

To nCMCC SPN, the expected number of tokens is the summation of the expected number of tokens for every place, tabbed with $\overline{N}$, thus

$$\overline{N} = \sum_{p_i \in S} \bar{u_i} \tag{3}$$

Average number of tokens is a very useful parameter for evaluating the performance of a model. We will give detail explanation later.

Based on Little Formula and principle of Balance, we can conclude the expected delay time of a subsystem. The result of T is important not only in evaluating the run time of nCMCC, but also to algorithms on clusters. Suppose our nCMCC is a subsystem, we can result the following formula.

Given T is the expected delay time of SPN, λ is the expected coming rate, thus

$$T = \overline{N}/\lambda \tag{4}$$

Based on the probability of steady state in Markov chain, we can result many other performance targets in the theory of SPN. Other performance targets can be found in [5−9]. And the nCCAM SPN can be converted to Generalized Stochastic Petri Nets(GSPN) and Stochastic high-level Petri Nets(SHLPN) as needed.

3.3 The State Space of Markov Chain

According to the SPN model constructed above, the state space exponentially increases along with the increasing of n. How to reduce the explosion of the state space is a research problem in the theory of SPN, which is beyond our discussion.

Here we present a way to reduce the state space for special dimensions. This method is depended on the rules of nCMCC, not a general method used in the theory of SPN. Other methods based on GSPN and SHLPN are the future work of our research group.

Theorem 2. *Given a nCMCC and suppose the communication traffic to each node on clusters is the same, $a_i \in Va$, if $\bar{a_i} \in Va$, the behaviors of a_i and $\bar{a_i}$ are the same.*

Proof. Assume $a_i = (x_1 x_2 \cdots x_n)$, $\bar{a_i}=(\bar{x_1}\bar{x_2}\cdots\bar{x_n})$, where x_i and $\bar{x_i}$ are binary numbers. For $\forall a_i$ and $\bar{a_i} \in$ Va, there are no communication between a_i and $\bar{a_i}$. According to the *communication rule* of **definition 1**. At the i^{th} (i <=n) communication period, the objects of a_i and $\bar{a_i}$ are $(x_1 x_2 \cdots \bar{x_i} \cdots x_n)$ and $(\bar{x_1}\bar{x_2}\cdots x_i \cdots \bar{x_n})$ respectively, which are complementary bit by bit. Since the communication traffic to each node on clusters is the same, it is reasonable to say every communication process costs the same time. That is, every pair of nodes with complementary coding begin and finish the communication at the same time. So we can say the behaviors of a_i and $\bar{a_i}$ are the same.

Taking 4CMCC as an example, we can find four pairs of complementary nodes in Va. Based on theorem 2, the behaviors of these pairs of nodes are the same.

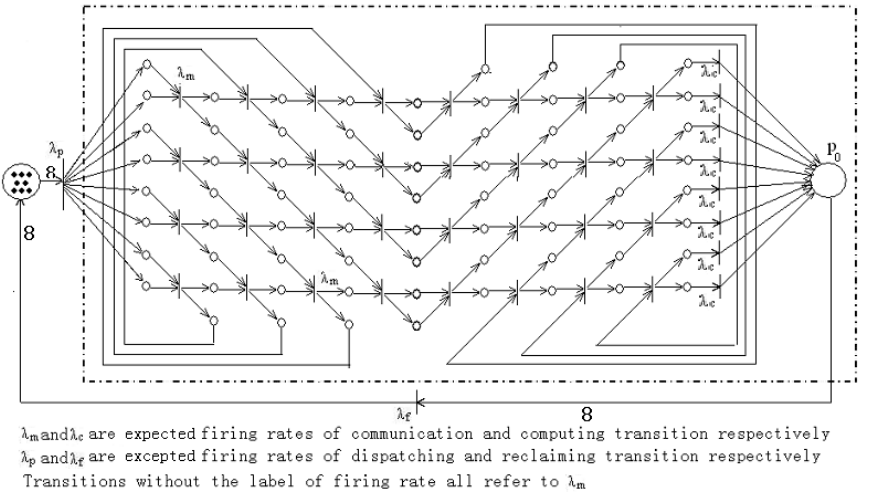


Fig. 3. SPN of 4CMCC after reducing state space. It acts as a subnet in the dashed line.

Thus, it is obvious that Va can be divided into two parts, which behave the same. This is the way to reduce the state space for this circumstance. Figure 3 gives the a sub SPN for 4CMCC, which is isomorphic with the whole SPN.

4 Case Study

The PSU/NCAR mesoscale model (known as MM5) is a limited-area, nonhydrostatic, terrain-following sigma-coordinate model designed to simulate or predict mesoscale atmospheric circulation.[10−12] The model is supported by several pre- and post-processing programs, which are referred to collectively as the MM5 modeling system. The core algorithm of MM5 is implemented using parallelism computing and mass data exchanging.[10] Currently, most of MM5 implementation are based on the cluster system.

Evaluating the performance of MM5 algorithm on cluster systems is not only a research problem but meaningful for practical usage. Traditional method is to add some functions for evaluation to the implemental software and gather the statistics of running time. The method needs plenty of time but can't evaluate the algorithm's performance from the model perspective.

The core algorithm of MM5 can be considered as iterative operating on several small rectangles which are evenly partitioned from the whole domain. The implemental algorithm for cluster system dispatches neighbor rectangles to logical neighbor nodes in the cluster. It is just the kind of algorithms which nCMCC aims at. MM5 model can be implemented using our communication model of any dimension. The main difference is how to dispatch the raw data and reclaim the results among computers, which won't influence the essence of the algorithm. In

this section we will analyze 4CMCC and its SPN to further evaluate the performance from the theory essence, which also makes sense to the algorithm of MM5.

4.1 Construction of 4CMCC and Its SPN

In a 4-cube, the nodes are encoded from 0000 to 1111 according to the *node rule* of **definition 1**. Figure 1 gives the graphic description of this 4CMCC.

According to the analysis of last section, we know that there is a better way to construct the SPN for reducing the explosion of state space. Using this method of reducing the state space, the Stochastic Petri Net constructed from 4CMCC is shown as a subnet in figure 3, which can be represented as {S, T; F, W, M_0, 4, λ}.

- S' = $\{p_0, p_1, \cdots, p_{72}\}$;
- T' is the set of transitions which are denoted in figure 3;
- F are described by arcs in figure 3;
- The weight function of arcs is constant. W(f) = 1;
- for $p_i \in$S, $M_0(p_i) = 0$;
- $\lambda = \{\lambda_m, \lambda_c\}$. λ_c and λ_m are the expected firing rates of computing transition and communication transition respectively.

Due to the application of MM5, the whole SPN has more transitions and places. Here, the representation of 4CMCC is a subnet to the whole SPN. λ_p and λ_f are the expected firing rates of dispatching and reclaiming transitions respectively.

4.2 Analysis and Results

In cluster systems, communication and computing time depends on many factors such as the bandwidth, the topology, the software algorithms, the computing capacity, the number of the nodes and so on. These factors can be classified into two kinds. One is the application specific(such as the amount of data needed to exchange) and the other is hardware specific.

Suppose the bandwidth and the computing capacity of cluster systems are denoted by $\boldsymbol{Bw}$ and $\boldsymbol{Pf}$. The application's communication cost and computation cost are denoted by N(σ) and C(σ), where σ is the data scale of an application. To a certain application, N(σ) and C(σ) are the functions of data scale, which is the reasonable circumstance. In our analysis, we assume $\lambda_p = \lambda_f = 0.01$ (in figure 3), and λ_m as a proportional function of $\boldsymbol{Bw}/\mathrm{N}(\sigma)$, λ_c as a proportional function of $\boldsymbol{Pf}/\mathrm{C}(\sigma)$. That is:

$$\lambda_m = k_1 \times \boldsymbol{Bw}/N(\sigma) \tag{5}$$

$$\lambda_c = k_2 \times \boldsymbol{Pf}/C(\sigma) \tag{6}$$

there, k_1 and k_2 are considered as the constant parameters.

We take the software of *spnp-GUI* downloaded from the Duke University as our simulation tool in this section. It can output the expected toke number of

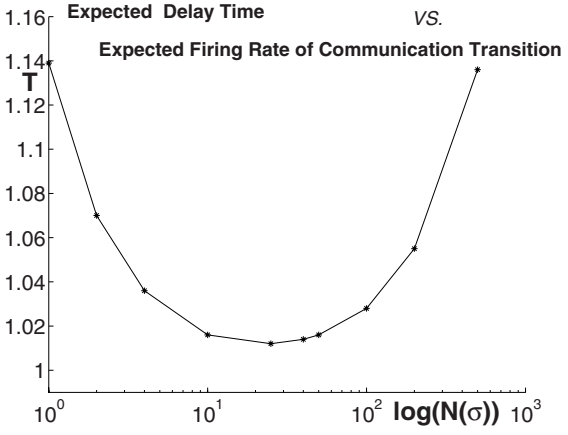


Fig. 4. Trade-off between computing capacity of nodes and the bandwidth

a given place in steady-state, the utilization for a given transition in steady-state, and so on. They are referred when computing the expected delay time of a sub-SPN.

According to the equations (5)(6), we can get

$$\boldsymbol{Bw} \times \boldsymbol{Pf} = \frac{N(\sigma) \times C(\sigma)}{k_1 \times k_2} \times (\lambda_m \times \lambda_c) \tag{7}$$

Since the network bandwidth and computing capacity of nodes on clusters are the most important two factors that determine the performance of the system, we can reasonably consider $\boldsymbol{Bw} \times \boldsymbol{Pf}$ as a measurement of the whole cluster systems. In equation (7), if $\frac{N(\sigma) \times C(\sigma)}{k_1 \times k_2}$ is a fixed value, $\lambda_m \times \lambda_c$ can reflect the cluster's whole performance too.

Given $\lambda_m \times \lambda_c$=100, the simulation result is showed in Figure 4. It implies that when the cluster's whole performance remains the same, the expected execution time of a given application decreases first and then increases with the bandwidth decreasing. It can be interpreted as following. When the computing capacity of nodes is relative slow to the bandwidth, the execution time is determined by the computing capacity of nodes. But with the increasing of the computing capacity, the bandwidth becomes the bottleneck finally. Based on our nCMCC, we can find the best trade-off between computing capacity of nodes and the bandwidth.

In figure 5, λ_m remains the same while λ_c increases. The expected delay time decreases distinctly at the first, but when it arrives a specific level, it decreases so lightly that we can take the level as the approximate final value. That is to say, given an application, when the bandwidth of the cluster remains the same, improving the computing capacity of nodes can decrease the execution time considerably until the network bandwidth becomes the bottleneck. Based on our nCMCC, we can find the modest computing capacity of nodes to a certain bandwidth.

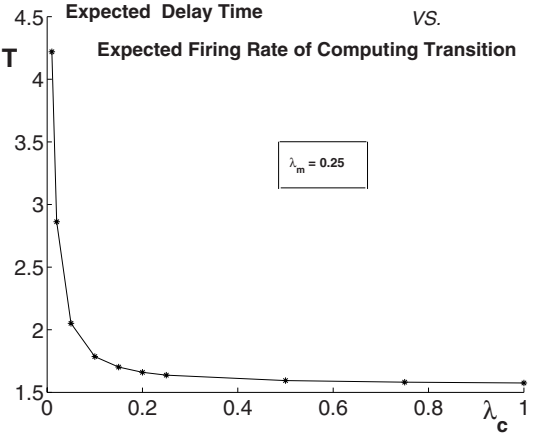


Fig. 5. The modest computing capacity of nodes to a certain bandwidth

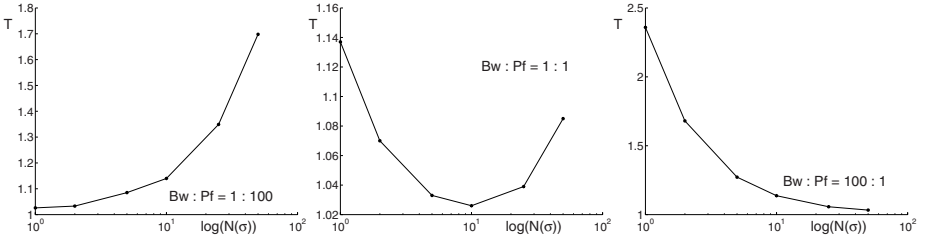


Fig. 6. Different partitions of the data exchanging and computing to three different systems

There may be several different algorithms to accomplish an application on the clusters. With the same essence of the algorithm, different partitions of the data exchanging and computing may lead to different communication and computing traffic, that is, different $N(\sigma)$ and $C(\sigma)$. It's reasonable to consider the product of $N(\sigma)$ and $C(\sigma)$ as a representation to an application. To a certain cluster system, a good algorithm has a partition consistent with the hardware specific factors. In figure 6, we try to find a theoretic law of partitions between the data exchanging and computing. Given three different proportions of $\boldsymbol{Bw}$ and $\boldsymbol{Pf}$, the delay time of the system may lead to different results along with the changing of $N(\sigma)$. The product of $N(\sigma)$ and $C(\sigma)$ is assumed to a fixed value of 50.

From figure 6, we can conclude that: when the algorithm leads to more communication than computing, the choice of a cluster with higher bandwidth than computing capacity of nodes is better. On the contrary, clusters with higher computing capacity than bandwidth have better performance when the computing is more than communication. Using our nCMCC, the kind of results can be shown easily and the quantitative results can be concluded.

From the last experiments, we get the similar conclusions with the traditional simulation. It shows that the model is correct and viable. Further more, compared with traditional method, some useful quantitative results can be evaluated more easily with the aid of SPN.

5 Conclusion and Future Work

It is an open problem to analyze and evaluate whether parallel algorithms take good advantage of the computing and networking resources of the clusters.[1−3] In this paper, we present a novel mathematic model nCMCC out of algorithms commonly used on clusters, referring to the same topology essence of mostly parallel algorithms.

This research is a contribution to the emerging area of performance evaluation of clusters. A SPN model of 4CMCC is built for MM5, and the performance of algorithms on different clusters is analyzed through three sets of experiments. It comes to a conclusion that good algorithms should base on the specific cluster's configuration to make best use of the cluster's resources. So the configuration of the network bandwidth and process frequency must be taken into account carefully.

Based on nCMCC, we can evaluate an algorithm's performance on the cluster with SPN easily. Further study for nCMCC will be processed using Stochastic High-Level Petri Nets(SHLPN) due to the exponential increasing state space of nCMCC's SPN.

References

1. Koibuchi, M., Watanabe, K., Kono, K., Jouraku, A., Amano, H.: Performance evaluation of routing algorithms in RHiNET-2 cluster. In: Cluster Computing. Proceedings 2003 IEEE International Conference, pp. 395–402 (2003)
2. Bessonov, O., Fougere, D., Roux, B.: Using a parallel cfd code for evaluation of clusters and MPPs. In: Parallel and Distributed Processing Symposium, Proceedings 2003 International, pp. 65–72 (2003)
3. Nguyen, K.N., Le, T.T.: Evaluation and comparison performance of various MPI implementations on an OSCAR linux cluster. In: Information Technology: Coding and Computing [Computers and Communications]. Proceedings. ITCC 2003 International Conference, pp. 310–314 (2003)
4. Xu, K., Fan, X.-b., Lin, C., Wu, J.-p.: Performance model and analysis of a distributed router. In: Communications, Circuits and Systems. IEEE 2002 International Conference, vol. 1, pp. 786–790 (2002)
5. Lin, C.: Performance evaluation of the computer network and computer system. Tsinghua University Press (2001)
6. Lin, C., Marinescu, D.C.: Stochastic high-level Petri nets and applications. IEEE Transactions on Computers 37(7), 815–825 (1988)
7. Lin, C., Qu, Y., Ren, F., Marinescu, D.C.: Performance Equivalent Analysis of Workflow Systems Based on Stochastic Petri Net Models. In: Proceedings of the First International Conference on Engineering and Deployment of Cooperative Information Systems (2002)

8. Molloy, M.K.: Performance Analysis Using Stochastic Petri Nets. IEEE Trans. Comp. C-39(9), 913–917 (1982)
9. Lopez-Benitez, N.: Dependability analysis of distributed computing systems using stochastic Petri nets. Reliable Distributed Systems, pp. 85 - 92 (1992)
10. Grell, G.A., Dudhia, J., Stauffer, D.R.: A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5). NCAR Technical Note NCAR/TN-398+STR, (June 1994)
11. Michalakes, J: The same source parallel MM5. In: Proceedings Second International Workshop on Software Engineering and Code Design in Parallel Meteorological and Oceanographic Applications 1998, Greenbelt, MD, USA.
12. MM5 Homepage, http://www.mmm.ucar.edu/mm5/mm5-home.html
13. Ming-Yun, H.: Analysis of Boolean N-cube interconnection networks for multiprocessor systems. Doctoral Dissertation
14. Almeida, V.A.F., Dowdy, L.W., Leuze, M.R.: An analytic model for parallel Gaussian elimination on a binary N-Cube architecture. In: Proceedings of the third conference on Hypercube concurrent computers and applications, pp. 1550 - 1553 (1989)
15. Yang, C.S., Wang, J.F., Lee, J.Y., Boesch, F.T.: Graphic Theoretic Reliability Analysis for the Boolean n cube networks. IEEE Transactions on circuits and systems 35(9), 1175–1179 (1988)

An Algorithm to Find Optimal Double-Loop Networks with Non-unit Steps

Xiaoping Dai[1], Jianqin Zhou[1], and Kaihou Wang[2,⋆]

[1] Department of Computer Science, Anhui University of Technology
Ma'anshan, 243002 China
[2] Department of Mathematics, Shijiazhuang University of Economics
Shijiazhuang, 050031 China
xpdai@ahut.edu.cn, zhou9@yahoo.com,
khwang@sjzue.edu.cn

Abstract. A double-loop network(DLN) $G(N;r,s)$ is a digraph with the vertex set $V=\{0,1,\ldots,N-1\}$ and the edge set $E=\{v\rightarrow v+r(\mod N)$ and $v\rightarrow v+s(\mod N)|v\in V\}$. Let $D(N;r,s)$ be the diameter of G, $D(N)=\min\{D(N;r,s)|1\leq r<s<N$ and $\gcd(N;r,s)=1\}$ and $D_1(N)=\min\{D(N;1,s)|1<s<N\}$. Although the identity $D(N)=D_1(N)$ holds for infinite values of N, there are also another infinite set of integers with $D(N)<D_1(N)$. These other integral values of N are called non-unit step integers or nus integers. Xu and Aguiló et al. gave some infinite families of 0-tight nus integers with $D_1(N)-D(N)\geq 1$.

In this work, an algorithm is derived for finding nus integers. The running time complexity of the proposed algorithm is $O(k^2)O(N^{1/4}\log N)$. It is verified by computer that the algorithm works extremely well. A new approach is also proposed for finding infinite families of nus integers. As an example, we present an infinite family of of 0-tight nus integers with $D_1(N)-D(N)=4$.

Keywords: Double-loop network, tight optimal, L-shaped tile, non-unit step integer, algorithm.

1 Introduction

Double-loop digraphs $G=G(N;r,s)$, with $1\leq r<s<N$ and $\gcd(N;r,s)=1\}$, have the vertex set $V=\{0,1,\ldots,N-1\}$ and the adjacencies are defined by $v\rightarrow v+r(\mod N)$ and $v\rightarrow v+s(\mod N)$ for $v\in V$. These kinds of digraphs have been widely studied as architectures for local area networks, known as double-loop networks (DLN). For surveys about these networks, refer to [3,7].

From the metric point of view, the minimization of the diameter of G corresponds to a faster transmission of messages in the network. The diameter of G is denoted by $D(N;r,s)$. As G is vertex symmetric, its diameter can be computed from the expression $\max\{d(0;i)|i\in V\}$, where $d(u;v)$ is the distance from u to

⋆ The research was supported by Chinese Natural Science Foundation (No. 60473142).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 352–361, 2007.

v in G. For a fixed integer $N > 0$, the optimal value of the diameter is denoted by

$$D(N) = \min\{D(N; r, s) | 1 \leq r < s < N \text{ and } \gcd(N; r, s) = 1\}$$

Several works studied the minimization of the diameter (for a fixed N) with $r = 1$. Let us denote

$$D_1(N) = \min\{D(N; 1, s) | 1 < s < N\}$$

Since the work of Wong and Coppersmith [10], a sharp lower bound is known for $D_1(N)$:

$$D_1(N) \geq \lceil \sqrt{3N} \rceil - 2 = lb(N)$$

Fiol et al. in [8] showed that $lb(N)$ is also a sharp lower bound for $D(N)$. A given DLN $G(N; r, s)$ is called k-tight if $D(N; r, s) = lb(N) + k (k \geq 0)$. A k-tight DLN is called optimal if $D(N) = lb(N) + k (k \geq 0)$, hence integer N is called k-tight optimal. The 0-tight DLN are known as tight ones and they are also optimal. A given DLN $G(N; 1, s)$ is called k-tight if $D(N; 1, s) = lb(N) + k (k \geq 0)$. A k-tight DLN is called optimal if $D_1(N) = lb(N) + k (k \geq 0)$.

The metrical properties of $G(N; r, s)$ are fully contained in its related L-shaped tile $L(N; l, h, x, y)$, where $N = lh - xy, l > y$ *and* $h \geq x$. In Figure 1, we illustrate generic dimensions of an L-shaped tile.

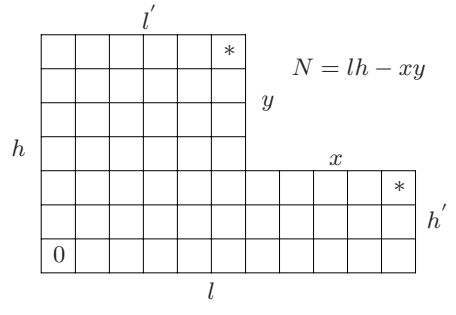


Fig. 1. Generic dimensions of an L-shaped tile

Let $D(L) = D(L(N; l, h, x, y)) = \max\{l + h - x - 2, l + h - y - 2\}$. For obvious reasons, the value $D(L)$ is called the diameter of the tile L. It is known that an L-shaped tile $L(N; l, h, x, y)$ can be assigned to a $G(N; r, s)$ without any confusion. However, we can not find double-loop network $G(N; r, s)$ from some L-shaped tiles. When an L-shaped tile $L(N; l, h, x, y)$ has diameter $lb(N) + k$, we say it is k-tight.

Xu [11] presented three infinite families of 0-tight nus integers with $D_1(N) - D(N) \geq 1$. Aguiló et al. [2] derived a method for finding infinite families of nus integers and then presented some infinite families of 0-tight nus integers with $D_1(N) - D(N) \geq 1$. It is known that finding infinite families of nus integers with $D_1(N) - D(N) \geq k$ is a difficult task as the value k increases.

The remaining of this paper will be organized as follows. Some lemmas, which will be used throughout this paper, are introduced in Section 2. In section 3, we propose a simple and efficient algorithm to search an L-shaped tile $L(N; l, h, x, y)$ with diameter $\lceil\sqrt{3N}\ \rceil - 2 + k$ in the order $k = 0, 1, 2, \cdots$. Our algorithm is based on some theorems of Li and Xu[9,11,12]. The running time complexity of our algorithm is $O(k^2)O(N^{1/4}\log N)$. Based on the algorithm to search L-shaped tiles, we give an algorithm for finding nus integers. The running time complexity of the algorithm is also $O(k^2)O(N^{1/4}\log N)$. Experiments show that the algorithm is fast and easy to realize. In section 4, an approach is derived to construct infinite families of nus integers. Finally, section 5 presents an infinite family of 0-tight nus integers with $D_1(N) - D(N) = 4$.

2 Preliminary

The following Lemma 1, 2, 3 and 4 can be found in [6 or 8 or 9].

Lemma 1[6,9]**.** Let t be a nonnegative integer. We define $I_1(t) = [3t^2+1, 3t^2+2t]$, $I_2(t) = [3t^2 + 2t + 1, 3t^2 + 4t + 1]$ and $I_3(t) = [3t^2 + 4t + 2, 3(t + 1)^2]$. Then we have $[4, 3T^2 + 6T + 3] = \bigcup_{t=1}^{T}\bigcup_{i=1}^{3} I_i(t)$, where $T > 1$, and $lb(N) = 3t + i - 2$ if $N \in I_i(t)$ for $i = 1, 2, 3$.

Lemma 2[8,11]**.** Let $L(N; l, h, x, y)$ be an L-shaped tile, $N = lh - xy$. Then
(a) There exists $G(N; 1, s)$ realizing the L-shaped tile iff $l > y$, $h \geq x$ and $\gcd(h, y) = 1$, where $s \equiv \alpha l - \beta(l - x)(\mod N)$ for some integral values α and β satisfying $\alpha y + \beta(h - y) = 1$.
(b) There exists $G(N; s_1, s_2)$ realizing the L-shaped tile iff $l > y$, $h \geq x$ and $\gcd(l, h, x, y) = 1$, where $s_1 \equiv \alpha h + \beta y(\mod N)$, $s_2 \equiv \alpha x + \beta l(\mod N)$ for some integral values α and β satisfying $\gcd(N, s_1, s_2) = 1$.

Lemma 3[9]**.** Let $L(N; l, h, x, y)$ be an L-shaped tile, $N = lh - xy$. Then
(a) If $L(N; l, h, x, y)$ is realizable, then $|y - x| < \sqrt{N}$;
(b) If $x > 0$ and $|y - x| < \sqrt{N}$, then
$D(L(N; l, h, x, y)) \geq \sqrt{3N - \frac{3}{4}(y - x)^2} + \frac{1}{2}|y - x| - 2$;
(c) Let $f(z) = \sqrt{3N - \frac{3}{4}z^2} + \frac{1}{2}z$. Then $f(z)$ is strictly increasing when $0 \leq z \leq \sqrt{N}$.

Lemma 4[9]**.** Let $N(t) = 3t^2 + At + B \in I_i(t)$ and L be the L-shaped tile $L(N(t); l, h, x, y)$, where A and B are integral values; $l = 2t + a$, $h = 2t + b$, $z = |y-x|$, a, b, x, y are all integral polynomials of variable t, and $j = i+k(k \geq 0)$. Then L is k-tight iff the following identity holds,

$$(a + b - j)(a + b - j + z) - ab + (A + z - 2j)t + B = 0. \tag{1}$$

The following Lemma 5 is the generalization of Theorem 2 in [12], and can be found in [13].

Lemma 5[13]**.** Let $H(z,j) = (2j-z)^2 - 3[j(j-z) + (A+z-2j)t + B]$, and the identity (1) be an equation of a and b. A necessary condition for the equation (1) to have integral solution is that $4H(z,j) = s^2 + 3m^2$, where s and m are integers.

Proof. Suppose that the equation (1) of variable a and b has an integral solution and rewrite it as the following,

$a^2 + (b - 2j + z)a + b^2 - (2j - z)b + c = 0$

where $c = j(j-z) + (A+z-2j)t + B$. Thus, there exists an integer m such that

$(b - 2j + z)^2 - 4[b^2 - (2j - z)b + c] = m^2$

Rewrite it as an equation of variable b,

$3b^2 - 2(2j - z)b + 4c + m^2 - (2j - z)^2 = 0$

Thus, there exists an integer n such that

$4(2j - z)^2 - 12[4c + m^2 - (2j - z)^2] = n^2$

This implies that n is even. Let $n = 2s$.

We have $4(2j - z)^2 - 12c = s^2 + 3m^2$, hence $4H(z,j) = s^2 + 3m^2$.

We have this lemma. □

It is easy to show that the following Lemma 6 is equivalent to Theorem 1 in [12]. Lemma 6 can be found in [13].

Lemma 6[13]**.** Let n, s and m be integers, $n = s^2 + 3m^2$. If n has a prime factor p, where $p \equiv 2 \pmod 3$, then there exists an even integer q, such that n is divisible by p^q, but not divisible by p^{q+1}.

Lemma 7[13]**.** Let $N = N(t) = 3t^2 + At + B \in I_i(t)$ and the L-shaped tile $L(N; l, h, x, y)$ be k-tight($k \geq 0$) and realizable. Let $z = |y - x|$. Then the following hold,

Case 1. If $A = 0$ or $A = 2$(if $i = 2$) or $A = 4$(if $i = 3$), and $3N - \frac{3}{4}(2k+3)^2 > (3t + \frac{A-1}{2})^2$, then $0 \leq z \leq 2k + 2$.

Case 2. If $A = 1$ or $A = 3$ or $A = 5$, and $3N - \frac{3}{4}(2k+2)^2 > (3t + \frac{A-1}{2})^2$, then $0 \leq z \leq 2k + 1$.

Case 3. If $A = 2$(if $i = 1$) or $A = 4$(if $i = 2$) or $A = 6$, and $3N - \frac{3}{4}(2k+1)^2 > (3t + \frac{A-1}{2})^2$, then $0 \leq z \leq 2k$.

Proof. We only prove Case 1. The others are similar.

Let $L(N; l, h, x, y)$ be k-tight. Then $D(L) = 3t + i - 2 + k$.

Note that $i = A/2 + 1$, by Lemma 3, if $z \geq 2k + 3$, we have

$$\begin{aligned} D(L(N; l, h, x, y)) &\geq \sqrt{3N(t) - \tfrac{3}{4}(2k+3)^2} + \tfrac{2k+3}{2} - 2 \\ &> (3t + \tfrac{A-1}{2}) + \tfrac{2k+3}{2} - 2 \\ &= 3t + i - 2 + k \\ &= lb(N) + k. \end{aligned}$$

Therefore, all k-tight L-shaped tile $L(N;l,h,x,y)$ must satisfy $0 \le z \le 2k+2, z = |y-x|$.

We have this lemma. □

Here we must note that the conditions of Lemma 7 are satisfied by almost all k-tight optimal and realizable L-shaped tile $L(N(t);l,h,x,y)$. If an L-shaped tile $L(N(t);l,h,x,y)$ does not satisfy the condition in Case 3. That is,

$3N(t) - \frac{3}{4}(2k+1)^2 \le (3t + \frac{A-1}{2})^2$

Hence,

$3(t+B) \le \frac{3}{4}(2k+1)^2 + (\frac{A-1}{2})^2$.

We may let $A_1 = A-1$ and $B_1 = B+t$. Then this is Case 2($i = \frac{A+1}{2}$) and probably the following holds,

$3(t+B_1) > \frac{3}{4}(2k+2)^2 + (\frac{A_1-1}{2})^2$.

This is equivalent to the condition in Case 2.

Otherwise, we may let $A_2 = A_1 - 1 = A-2$ and $B_2 = B_1 + t = B + 2t$. Then this is Case 1($i = \frac{A}{2}+1$) and probably the following holds,

$3(t+B_2) > \frac{3}{4}(2k+3)^2 + (\frac{A_2-1}{2})^2$.

This is equivalent to the condition in Case 1, to which there is no counter example up to now.

3 An Algorithm to Find nus Integers

Based on computer search, we know that,

The first N with 3-tight optimal double loop digraph is $3316 = 3\times 33^2+33+16$.

The first N with 4-tight optimal double loop digraph is $53749 = 3\times 133^2 + 5\times 133 + 17$.

The first N with 5-tight optimal double loop digraph is $417289 = 3\times 372^2 + 5\times 372 + 277$.

The first N with 6-tight optimal double loop digraph is $7243747 = 3\times 1553^2 + 5\times 1553 + 555$.

The first N with 7-tight optimal double loop digraph is $81190689 = 3\times 5202^2 + 5202 + 3075$.

Observe the above search results, we know that

Theorem 1. For $N \le 10^8$, if N with k-tight optimal double loop digraph, then $k < \log_{10} N$.

We continue to consider the case 1 of Lemma 7. The others are similar.

For $N(t) = 3t^2+At+B \in I_i(t)$, $3N(t) - \frac{3}{4}(2k+3)^2 > (3t+\frac{A-1}{2})^2$ is equivalent to that $3t + 3B > \frac{3}{4}(2k+3)^2 + \frac{1}{4}(A-1)^2$.

For $33 \le t < 372$, then $k \le 4$.

Given $A = 0, 2, 4$, then $\frac{3}{4}(2k+3)^2 + \frac{1}{4}(A-1)^2 < \frac{3}{4}(2\times 4+3)^2 + \frac{1}{4}\times 3^2 = 93 < 3\times 33 + 3 \le 3t + 3B$.

Consider $t = O(N^{1/2})$, t increases much faster than k^2 does, therefore,

$3t + 3B > \frac{3}{4}(2k+3)^2 + \frac{1}{4}(A-1)^2$ holds for $33 \le t \le 6000$ and $A = 0, 2, 4$.

For $t < 33$, we can directly verify that $0 \le z \le 2k+2$ holds. Based on Theorem 1, we have the following conclusion.

Theorem 2. For $N \leq 10^8$, let $N(t) = 3t^2+At+B \in I_i(t)$, where $t = \lceil \sqrt{N/3} \rceil - 1$, $A = \lfloor (N-3t^2)/t \rfloor$, $B = N - 3t^2 - At \geq 0$. If $D(N) = lb(N) + k(k \geq 0)$ and L-shaped tile $L(N; l, h, x, y)$ is k-tight, $z = |y - x|$, then the following holds,

Case 1. If $A = 0$ or $A = 2(i = 2)$ or $A = 4(i = 3)$, then $0 \leq z \leq 2k + 2$;

Case 2. If $A = 1$ or $A = 3$ or $A = 5$, then $0 \leq z \leq 2k + 1$;

Case 3. If $A = 2(i = 1)$ or $A = 4(i = 2)$ or $A = 6$, then $0 \leq z \leq 2k$.

The numerical computations and the results of Coppersmith reported in [5] suggest that the order of k might be as low as $O(\log^{1/4} N)$. Therefore, Theorem 1 and Theorem 2 might be also true for $N > 10^8$.

We can now describe our algorithm to search nus integers based on Theorem 2.

Algorithm 1. To check whether N is a nus integer.

Step 1. Given N, calculate: $t = \lceil \sqrt{N/3} \rceil - 1$; $A = \lfloor (N - 3t^2)/t \rfloor$; $B = N - (3t^2+At) \geq 0$; $lb(N) = \lceil \sqrt{3N} \rceil - 2$; $i = lb(N) - 3t + 2$; $k = 0$; $flag = 0, flag1 = 0$.

Case 1. If $A = 0$ or $A = 2(i = 2)$ or $A = 4(i = 3)$, then $z0 = 2$;

Case 2. If $A = 1$ or $A = 3$ or $A = 5$, then $z0 = 1$;

Case 3. If $A = 2(i = 1)$ or $A = 4(i = 2)$ or $A = 6$, then $z0 = 0$.

Step 2. Given k, then for $0 \leq z \leq 2k + z0$, look for k-tight tile $L(l, h, x, y)$ in the following order:

Loop: while(TRUE)

$j = i + k$;

Loop: for$(z = 0; z < 2k + z0 + 1; z = z + 1)$

If the equation $(a+b-j)(a+b-j+z) - ab + (A+z-2j)t + B = 0$ has integral solution (a, b), and

if(gcd$(2t + a, 2t + b, t + a + b - j, t + a + b - j + z) == 1$)

Begin

flag=1;

if (gcd$(2t+a, t+a+b-j) = 1$ or gcd$(2t+a, t+a+b-j+z) = 1$ or gcd$(2t+b, t+a+b-j) = 1$ or gcd$(2t+b, t+a+b-j+z) = 1$) then flag1=1 and break these two loops.

End

End

If flag=1, then break the while loop.

$k = k + 1$;

End

Step 3. There exists a k-tight(optimal) digraph $G(N; r, s)$. If flag1=0, then N is a nus integer; and if flag1=1, then N is not a nus integer. The algorithm ends.

If $D(N) = lb(N) + k(k \geq 0)$, then there exists k-tight L-shaped tiles , from Lemma 4 and Lemma 2, equation (1) has integral solution (a, b), such that $L(N; l, h, x, y)$ can be realized by $G(N; r, s)$, where $l = 2t + a, h = 2t + b, x = t + a + b - j, y = t + a + b - j + z$, and gcd$(l, h, x, y) = 1$. From Theorem 2, it is only need to search k-tight L-shaped tiles for $0 \leq z \leq 2k + z0$, where $z0$ is obtained from A and i.

If N is a nus integer, from Lemma 2, then $\gcd(l,x) > 1$, $\gcd(l,y) > 1$, $\gcd(h,x) > 1$ and $\gcd(h,y) > 1$, where $l = 2t + a, h = 2t + b, x = t + a + b - j, y = t + a + b - j + z$, for any integral solution (a,b) of equation (1) with $k = D(N) - lb(N)$, $0 \le z \le 2k + z0$. Therefore, Algorithm 1 is correct.

For the equation: $(a+b-j)(a+b-j+z) - ab + (A+z-2j)t + B = 0$. It can be rewritten as the following,

$$\frac{a^2+b^2}{2} + (\frac{a+b}{\sqrt{2}} + \frac{-2j+z}{\sqrt{2}})^2 = \frac{(-2j+z)^2}{\sqrt{2}} + j(z-j) + (2j-z-A)t - B$$

Note that $j = i + k$, and $0 \le z \le 2k + 2$, hence $b^2 \le 2[2j^2 + j(k+2-i) + (2j-A)t - B]$.

To solve the equation, we only need to check whether the quadratic equation $(a+b-j)(a+b-j+z) - ab + (A+z-2j)t + B = 0$ relating a has integral solution, where integral $b \in [-b_0, b_0]$, $b_0 = 2^{1/2}[2j^2 + j(k+2-i) + (2j-A)t - B]^{1/2}$.

Note that $t = O(N^{1/2})$, $B = O(N^{1/2})$, hence $b = O(N^{1/4})$, so the computing cost for searching integral pair (a,b) is $O(k^2)O(N^{1/4})$.

Next we have to check that $\gcd(2t+a, 2t+b, t+a+b-j, t+a+b-j+z) = 1$. It is well known that the order of the Euclidean algorithm to compute such a gcd is $O(\log N)$. Therefore, the computing cost for checking whether N is a nus integer is $O(k^2)O(N^{1/4} \log N)$.

2814 (found by computer search [2]) is the first nus integer which is related to 1-tight optimal digraph. With our algorithm, we have found that,

9306759 is a nus integer with $D_1(N) - D(N) = 4$,

539084 is the first nus integer which is related to 2-tight optimal digraph, and

36186111 is the first nus integer which is related to 3-tight optimal digraph.

4 An Approach to Generate Infinite Families of nus Integers

We now describe our approach to generate infinite families of nus integers.

Step 1. Find an integer N_0 , such that $G(N_0; s_1, s_2)$ is k-tight optimal ($k \ge 0$), and $D(N_0) < D_1(N_0)$.

Step 2. Find a polynomial $N(t) = 3t^2 + At + B$, such that $N(t_0) = N_0$ and $N(t) \in I_i(t)$,$1 \le i \le 3$.

Step 3. For any $H(z,j)$, $i \le j \le k$, $0 \le z \le 2k + z0$, where if $A = 0$ or $A = 2(i = 2)$ or $A = 4(i = 3)$, then $z0 = 2$; if $A = 1$ or $A = 3$ or $A = 5$,then $z0 = 1$; if $A = 2(i = 1)$ or $A = 4(i = 2)$ or $A = 6$, then $z0 = 0$.

Case 1. If $4H(z,j)$ does not have the form of $s^2 + 3m^2$, where s and m are integers. From Lemma 6, when $t = t_0$, $4H(z,j)$ has a prime factor $p \equiv 2(\mod 3)$, and there exists an even integer q, such that $4H(z,j)$ is divisible by p^{q-1} , but not divisible by p^q. Suppose we have got the following factors:

$p_1^{q_1}, p_2^{q_2}, \ldots, p_l^{q_l}$.

Let $g_0 = lcm(p_1^{q_1}, p_2^{q_2}, \ldots, p_l^{q_l})$.

Case 2. If $j = k$, $A + z - 2j = 0$, and $4H(z, j)$ has the form of $s^2 + 3m^2$, where s and m are integers. For any integral solution (a, b) of the equation: $(a + b - j)(a + b - j + z) - ab + (A + z - 2j)t + B = 0$.

Let $l(t) = 2t + a = 2(t - t_0) + l_0$, $h(t) = 2t + b = 2(t - t_0) + h_0$, $x(t) = t + a + b - j$(or $y(t) + z) = (t - t_0) + x_0$, $y(t) = x(t) + z$ (or $(t + a + b - j) = (t - t_0) + y_0$. From Lemma 1, as $L(N; l, h, x, y)$ can not be realized by $G(N; 1, s)$, hence $\gcd(h_0, y_0) > 1$, so there exists a prime factor p of $\gcd(h_0, y_0)$. Suppose we have got the following prime factors:

$p_1, p_2, \ldots, p_r$.

Let $g_1 = lcm(p_1, p_2, \ldots, p_r)$.

Step 4. Suppose $\{G(N(t); s_1(t), s_2(t)) : t = g_2 e + t_0, e \geq 0\}$ is an infinite family of k-tight DLN(not necessarily optimal), then $\{N(t) : t = ge + t_0, e \geq 0\}$, where $g = lcm(g_0, g_1, g_2)$, is an infinite family of k-tight nus integers.

From Step 3, we know that our method can only deal with a part of nus integers.

5 An Application Example

We now apply our approach to generate an infinite family of nus integers.

Example 1. Take $N(t) = 3t^2 + 2t - 126$, and $N(1761) = 9306759$.

For $D(9306759; 7, 5269) = lb(9306759) = 5282$, then $G(N; 7, 5269)$ is 0-tight optimal.

On the other side, for $D(9306759; 1, 161420) = 5286 = lb(N) + 4$, and it is checked by computer that $D_1(9306759) = D(9306759; 1, 161420) = 5286$, so 9306759 is a nus integer with $D_1(N) - D(N) = 4$. In fact, the following proof will show that $D_1(9306759) \geq 5286$.

For $A = 2$, $B = -126$, $j = 1$, $z = 0$, then $A + z - 2j = 0$, so the equation (1) becomes

$(a + b - 1)(a + b - 1) - ab - 126 = 0$,

which has integral solutions:

$S = \{(-9, -2), (-9, 13), (-2, -9), (-2, 13), (13, -9), (13, -2)\}$.

For $(-9, -2)$, $gcd(h, y) = \gcd(2t + b, t + a + b - j) = \gcd(2t - 2, t - 12) = 11$ if $t \equiv 1(\mod 11)$.

For $(-9, 13)$, $gcd(h, y) = \gcd(2t + 13, t + 3) = 7$ if $t \equiv 4(\mod 7)$.

For $(-2, -9)$, $gcd(h, y) = \gcd(2t - 9, t - 12) = 3$ if $t \equiv 0(\mod 3)$.

For $(-2, 13)$, $gcd(h, y) = \gcd(2t + 13, t + 10) = 7$ if $t \equiv 4(\mod 7)$.

For $(13, -9)$, $gcd(h, y) = \gcd(2t - 9, t + 3) = 3$ if $t \equiv 0(\mod 3)$.

For $(13, -2)$, $gcd(h, y) = \gcd(2t - 2, t + 10) = 11$ if $t \equiv 1(\mod 11)$.

By Lemma 2(a) and consider the symmetry of L-shaped tile, for $t = 3 \times 7 \times 11 \times e + 1761(e \geq 0)$, there is no $G(N; 1, s)$ realizing the 0-tight L-shaped tile $L(N(t); l, h, x, y)$ where $y = x$.

Some L-shaped tiles can not be realized by $G(N; 1, s)$, but can be realized by $G(N; s_1, s_2)$.

Take $j = 1$, $z = 0$, $(a, b) = (-2, 13) \in S$, let $l = 2t + a$, $h = 2t + b$, $x = t + a + b - j$, $y = x$, $\alpha = -1$, $\beta = 2$, $s_1 \equiv \alpha h + \beta y(\mod N) = 7$, $s_2 \equiv \alpha x + \beta l(\mod N) = 3t - 14$. Then for $t = 7e + 1761 (e \geq 0)$, $\gcd(l, h, x, y) = \gcd(2t - 2, 2t + 13, t + 10, t + 10) = 1$ and $\gcd(N, s_1, s_2) = 1$.

Hence, $\{G(N(t); s_1, s_2) : t = 7e + 1761, e \geq 0\}$ is an infinite family of 0-tight DLN.

For $2 \leq j \leq 4$, $0 \leq z \leq 2(j-1)$, $t = 1761$, $H(z, j)$ has the following factors:

$H(0, 2) = 10948 = 17 \times 644$, where the power of 17 is 1.
$H(1, 2) = 5664 = 32 \times 177$, where the power of 2 is 5.
$H(2, 2) = 382 = 2 \times 191$, where the power of 2 is 1.
$H(0, 3) = 21519 = 797 \times 27$, where the power of 797 is 1.
$H(1, 3) = 16234 = 2 \times 8117$, where the power of 2 is 1.
$H(2, 3) = 10951 = 47 \times 233$, where the power of 47 is 1.
$H(3, 3) = 5670 = 2 \times 2835$, where the power of 2 is 1.
$H(4, 3) = 391 = 17 \times 23$, where the power of 17 is 1.
$H(0, 4) = 32092 = 71 \times 452$, where the power of 71 is 1.
$H(1, 4) = 26806 = 2 \times 13403$, where the power of 2 is 1.
$H(2, 4) = 21522 = 2 \times 10761$, where the power of 2 is 1.
$H(3, 4) = 16240 = 5 \times 3248$, where the power of 5 is 1.
$H(4, 4) = 10960 = 5 \times 2192$, where the power of 5 is 1.
$H(5, 4) = 5682 = 2 \times 2841$, where the power of 2 is 1.
$H(6, 4) = 406 = 2 \times 203$, where the power of 2 is 1.

Let $g_0 = 2^6 \times 5^2 \times 17^2 \times 47^2 \times 71^2 \times 797^2$, and $t = 3 \times 7 \times 11 \times g_0 \times e + 1761 (e \geq 0)$.

For $t \geq 1761$, $A = 2$, $B = -126$, $0 \leq k \leq 3$, we know that $3N(t) - \frac{3}{4}(2k+1)^2 > (3t + \frac{A-1}{2})^2$ is equivalent to $3t + 3B > \frac{3}{4}(2k+3)^2 + \frac{1}{4}(A-1)^2$, which is true. From Lemma 7, if L-shaped tile $L(N; l, h, x, y)$ is k-tight, $z = |y - x|$, then $0 \leq z \leq 2k$. We know that there is no 0-tight L-shaped tile $L(N; l, h, x, y)$, $|y - x| = 0$, which can be realized by $G(N; 1, s)$, for $t = 3 \times 7 \times 11 \times g_0 \times e + 1761 (e \geq 0)$.

For $1 \leq k \leq 3 (2 \leq j \leq 4), 0 \leq z = |y - x| \leq 2k$, by Lemma 6, $H(z, j)$ does not have the form of $s^2 + 3m^2$. By Lemma 5, the equation (1) has no integral solutions of a and b. By Lemma 4, there is no k-tight L-shaped tile $L(N(t); l, h, x, y)$ for (z, j).

As a conclusion, the nodes $N(t) = 3t^2 + 2t - 126, t = 3 \times 7 \times 11 \times g_0 \times e + 1761 (e \geq 0)$, of an infinite family of 0-tight optimal DLN correspond to nus integers with $D_1(N) - D(N) \geq 4$.

We now derive further an infinite family of 0-tight optimal DLN corresponding to nus integers with $D_1(N) - D(N) = 4$.

It is easy to obtain that $l = 3612, h = 3517, x = 1841, y = 1845$ from 4-tight optimal DLN $G(9306759; 1, 161420)$. For $t = 1761$, we have $a = 90, b = -5, z = 4$. As $j = 1 + 4A = 2, B = -126, A + z - 2j = -4$, let $a = 90, b = 4f - 5$.

From the equation (1), we have $t = 4f^2 - 74f + 1761$. Let

$l(f) = 2t + a = 8f^2 - 148f + 3612$,
$h(f) = 2t + b = 8f^2 - 144f + 3517$,
$x(f) = t + a + b - j = 4f^2 - 70f + 1841$,
$y(f) = x(t) + z = 4f - 70f + 1845$,
$l'(f) = l(f) - x(f) = 4f^2 - 78f + 1771$,

$h'(f) = h(f) - y(f) = 4f^2 - 74f + 1672$.

It is easy to verify that $\gcd(y(f), h'(f)) = 1$ for $f = 49419g$, and there exit $\alpha(g) = 34289gf - 63422g + 29$ and $\beta(g) = -34289gf + 59994g - 32$, such that $\alpha(g)y(f) + \beta(g)h'(f) = 1$.

Note that $t - 1761 = f(4f - 74), 49419 = 17 \times 3 \times 969$, and if $(32 \times 17)|g$, then $(3 \times 64 \times 17^2)|f(7f - 66)$.

It follows that $N(t)$ is not k-tight$(0 \le k \le 3)$ optimal, where $N(t) = 3t^2 + 2t - 126, t = 4f^2 - 74f + 1761, f = 49419g, g = 7 \times 11 \times 32 \times 5^2 \times 17 \times 47^2 \times 71^2 \times 797^2 e, e \ge 0$.

By Lemma 2(a), we conclude that $G(N(t); 1, s(g))$ is 4-tight optimal DLN corresponding to nus integers with $D_1(N) - D(N) = 4$, where

$N(t) = 3t^2 + 2t - 126, t = 4f^2 - 74f + 1761$,
$s(g) = \alpha(g)l(f) - \beta(g)l'(f)$,
$l'(f) = l(f) - x(f) = 4f^2 - 78f + 1771$,
$\alpha(g) = 34289gf - 63422g + 29$,
$\beta(g) = -34289gf + 59994g - 32$,
$f = 49419g, g = 7 \times 11 \times 32 \times 5^2 \times 17 \times 47^2 \times 71^2 \times 797^2 e, e \ge 0$.

Suppose $e = 0$, then $f = g = 0, t = 1761, N(1761) = 9306759, s(0) = 29 \times 3612 + 32 \times 1771 = 161420$. Thus, $G(N(1761); 1, s(0))$ is a 4-tight optimal DLN corresponding to a nus integer with $D_1(N) - D(N) = 4$.

References

1. Aguiló, F., Fiol, M.A.: An efficient algorithm to find optimal double loop networks. Discrete Mathematics 138, 15–29 (1995)
2. Aguiló, F., Simó, E., Zaragozá, M.: Optimal double-loop networks with non-unit steps. The Electronic Journal of Combinatorics 10, ♯R2(2003)
3. Bermond, J.-C., Comellas, F., Hsu, D.F.: Distributed loop computer networks: a survey. J. Parallel Distribut. Comput. 24, 2–10 (1995)
4. Chan, C.F., Chen, C., Hong, Z.X.: A simple algorithm to find the steps of double-loop networks. Discrete Applied Mathematics 121, 61–72 (2002)
5. Erdös, P., Hsu, D.F.: Distributed loop networks with minimum transmission delay. Theoret. Comput. Sci. 100, 223–241 (1992)
6. Esqué, P., Aguiló, F., Fiol, M.A.: Double commutative-step digraphs with minimum diameters. Discrete Mathematics 114, 147–157 (1993)
7. Hwang, F.K.: A complementary survey on double-loop networks. Theoret. Comput. Sci. 263, 211–229 (2001)
8. Fiol, M.A., Yebra, J.L.A., Alegre, I., Valero, M.: A discrete optimization problem in local networks and data alignment. IEEE Trans. Comput. C-36, 702–713 (1987)
9. Li, Q., Xu, J., Zhang, Z.: The infinite families of optimal double loop networks. Discrete Applied Mathematics 46, 179–183 (1993)
10. Wong, C.K., Coppersmith, D.: A combinatorial problem related to multimode memory organizations. J. Ass. Comput. Mach. 21, 392–402 (1974)
11. Xu, J.: Designing of optimal double loop networks. Science in China, Series E E-42(5), 462–469 (1999)
12. Xu, J., Liu, Q.: An infinite family of 4-tight optimal double loop networks. Science in China, Series A A-46(1), 139–143 (2003)
13. Zhou, J., Xu, X.: On infinite families of optimal double-loop networks with non-unit steps, Ars Combinatoria (accepted)

Self-adaptive Adjustment on Bandwidth in Application-Layer Multicast*

Jianqun Cui[1,2], Yanxiang He[2], and Libing Wu[2]

[1] Department of Computer Science, Huazhong Normal University, Wuhan, 430079, China
[2] School of Computer, Wuhan University, Wuhan, 430072, China
jqcui@126.com,{yxhe,wu}@whu.edu.cn

Abstract. Traditionally, most ALM researches have been focusing on the connectivity among the hosts by addressing how messages are routed from one source to all the other group members. In fact lots of applications require a certain target bandwidth to provide fluent service. Some ALM protocols consider the bandwidth as one of the metrics when constructing the overlay. But these mechanisms just can guarantee the initial performance of ALM trees. The quality of the multicast service may be unavailable when conditions are changed. In this paper, a self-adaptive adjustment on bandwidth (SAB) mechanism is introduced to ensure certain bandwidths for certain applications during the whole ALM service phase. The main idea of our mechanism is to relieve the burden of parent nodes from overload before the congestion occurs. The bandwidth monitor and the tree adjustment algorithms are used to implement the mechanism. Simulation results show that the SAB mechanism can get higher performance than normal ALM protocols (such as TAG) in the unstable environment.

1 Introduction

IP Multicast [1] is the most efficient way to perform group data distribution, as it eliminates the traffic redundancy and improves the bandwidth utilization on the network. But today's Internet service providers are still reluctant to provide a wide-area multicast routing service due to some reasons such as forwarding state scalability, full router dependence and so on [2]. In order to overcome current limitations in IP multicast, application-layer multicast (ALM) has been proposed, where the group members form an overlay network and data packets are relayed from one member to another via unicast.

Traditionally, most ALM researches [3, 4, 5, 6, 7, 8] have been focusing on the connectivity among the hosts by addressing how messages are routed from one source to all the other group members. In fact lots of applications (such as network video meetings, live TV programs) require a certain target bandwidth to provide fluent service. Some ALM protocols [9,10,11] consider the bandwidth as one of the metrics

* Supported by the Important Research Plan of National Natural Science Foundation of China (No. 90104005).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 362–371, 2007.

when constructing the overlay. But these mechanisms just can guarantee the initial performance of ALM trees and ignore the unstable status of networks. The quality of multicast service may be unavailable when conditions are changed. Too many children being added to one parent node, network congestion or host over loading may lead the multicast application out of service in some nodes. The video streaming will be broken or frozen in these nodes. If the application-layer multicast mechanism can adjust the multicast tree structure ahead to balance the bandwidth of whole tree, the block situation will be avoided.

We present a self-adaptive adjustment mechanism on bandwidth (SAB) to ensure certain bandwidths for certain applications. Each group member keeps a bandwidth monitor to probe current bandwidth. The monitor computes bandwidth rank based on collected results and the bandwidth rank decides the out degree of multicast nodes. The SAB mechanism can move redundant children to other nodes if necessary to acquire enough bandwidth which can support the multicast application.

The paper is structured as follows. Section 2 provides an overview of related work and section 3 introduces the proposed self-adaptive adjustment mechanism. Performance of the mechanism is verified by simulation in section 4. Finally, we conclude the paper with a summary of conclusion and future work.

2 Related Work

Many application-layer multicast solutions have been put forward with their respective strengths and weaknesses [12]. More and more researchers take the bandwidth into account when building the overlay for the QoS of multicast application. TAG [10,11] uses the network topology information to construct an overlay network with low delay penalty and a limited number of identical packets. Bandwidth is also considered in tree construction as a secondary metric. TAG can meet the requirement of short physical link and certain bandwidth. FAT [9] scheme provides a method to build a high-bandwidth overlay tree on top of the inferred underlay. The experiment results shows FAT can achieve higher bandwidth than Narada [12] and Overcast [12]. Yang Zhong etc. proposed a proxy-based single source ALM protocol ProBass [13] which targets media streaming applications, where out degree constraint and end-to-end distance are two metrics. Out degree reflects the maximum bandwidth each node can provide.

All of solutions introduced above consider the bandwidth as one of the metrics to construct the overlay network. But they never check bandwidth changes when the multicast tree keeps relative stable (no nodes to join or leave). It means the number of children will not be decreased even if the parent has not enough bandwidth to support the application. Just when the children think the parent has left the tree, these children will join to other parents. In fact, the parent may not leave the tree and just can not afford so many children. So the wrong cognizance will lead more children to rejoin the tree and the parent will waste its bandwidth. To solve these problems, we propose a self-adaptive adjustment on bandwidth (SAB) mechanism that can adjust the number of children automatically based on the measured bandwidth.

3 Self-adaptive Adjustment on Bandwidth

The main idea of our SAB mechanism is to relieve the burden of the parent node from overload before the congestion occurs. Member join algorithms of most ALM protocols take the factor of the bandwidth into account. For example, choosing node with maximum bandwidth as new member's parent or limiting the out degree of a parent node is an efficient way to construct the overlay network. But members may have different available bandwidths during the whole phase of the multicast application for some reasons. So it is necessary to monitor and adjust the available bandwidth to adapt the change of the environment.

3.1 Bandwidth Rank and Out Degree

We classify the available bandwidth of multicast members into several ranks. For simplicity, we define the rank of node *i* as follows:

$$R_i = Int(Ba_i / Br) \tag{1}$$

Here, Ba_i denotes the available bandwidth of the node *i*. *Br* is the minimum required bandwidth of the multicast application. Function $Int(x)$ returns the whole number part of the number x without rounding. For example, assuming Ba_i is 2.5 Mbps and Br is 300 Kbps, the bandwidth rank of the node *i* will be 8. We use the bandwidth rank to limit the out degree of a multicast member. If the bandwidth rank of the node *i* is *n*, it means the node can add *n* more children into the multicast tree. So the bandwidth rank of a fully loaded node should be 0. The relationship between the bandwidth rank and the out degree is shown as follows:

$$Max(O_i) = R_i + Oc_i \tag{2}$$

$Max(O_i)$ denotes the maximum out degree of the node *i* and Oc_i is the current degree of the node *i*. If Oc_i reaches to $Max(O_i)$, R_i will be 0, which means no children should be added to the node *i*.

3.2 Bandwidth Monitor

The minimum required bandwidth *Br* is a constant for a given application. However, the available bandwidth is a variable and changed with time. Most of application-layer multicast protocols just concern the initial bandwidth of members. The initial bandwidth just can represent the available bandwidth when the node joins the tree. We need a monitor to watch the available bandwidth at intervals so that we can balance the tree as soon as possible.

As we all know, available bandwidth can be calculated as follows:

$$Bandwidth = Size / Transmit \tag{3}$$

Here *Size* refers to the size of a packet and *Transmit* is the delay of sending the packet. So we can construct a probe packet with a certain size and send it to one of its children to measure the approximate sending bandwidth.

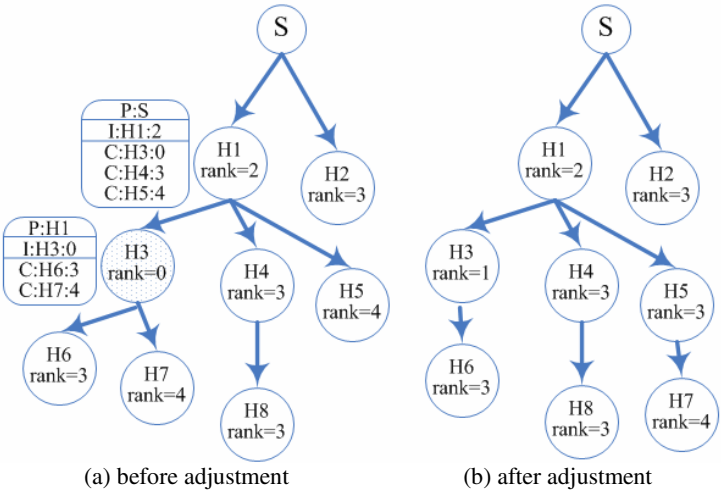
Bandwidth monitor creates the probe packet and transmits it in a certain interval. It records the transmit delay of the packet to get the current available bandwidth. Then the monitor use equation (1) to update the bandwidth rank of the node. If the rank is 0, bandwidth monitor will start the tree adjustment algorithm to abate the burden of the node. We can find through equation (1) that $R_i = 0$ does not mean there is no more available bandwidth. It just denotes that the available bandwidth can not afford one more child to request for service from the node. So our self-adaptive adjustment can balance the tree bandwidth before the congestion occurs. Here, *self-adaptive* means the bandwidth will be adjusted automatically with the variety of the environment. The information of the bandwidth rank will be uploaded to the parent for later use.

The Real-Time Control Protocol (RTCP) seems to do the similar job to the bandwidth monitor. In fact, they are difference. Our bandwidth monitor just need probe it own bandwidth and send the information to its parent while the receivers running on RTCP must send RTCP messages to all senders periodically.

The algorithm of the bandwidth monitor will be described in the next section.

3.3 Tree Adjustment Algorithm

Tree adjustment algorithm is used to adjust the number of children. It will be triggered when the bandwidth rank of the node is decreased to 0. The mechanism is illustrated in Fig. 1.



(a) before adjustment (b) after adjustment

Fig. 1. Tree Adjustment

Source *S* is the root of the multicast tree, and *H*1 through *H*8 are hosts in Fig. 1. Each node of the multicast tree maintains a family table (FT) defining parent-child relationship and their bandwidth rank. In the family table, *P* denotes the parent, *C* denotes the child and *I* denotes the node itself. For example, *C*:*H*4:3 (Fig. 1 (a)) means *H*4 is the child of the node and its bandwidth rank is 3. We do not need to record the bandwidth rank of the parent for the parent is unique.

In Fig. 1 (a), *H*3's bandwidth monitor finds its rank is 0 and the tree adjustment algorithm is triggered. *H*3 sends ADJUST message to its parent. The parent *H*1 will choose *H*5 as the new parent because *H*5's rank is highest among all of the children. When *H*3 receives the response message PARENT from *H*1, it will know that it can move one of its children to *H*5. Then it sends ADD message to *H*5 with the information of the child with maximum rank (*H*7). If *H*5 accepts *H*7 as its child, it will provide service to *H*7 right now and sends ADD_OK message to *H*3. *H*3 will remove *H*7 from the family table and stop transmitting data to *H*7 after it receives the ADD_OK message. The adjustment result is shown in Fig. 1 (b).

We choose one of the neighbors as the new parent because it can keep the same depth of whole multicast tree and neighbors have certain relationship (such as close to each other) in most application-layer multicast protocols. We can also choose other child (for example *H*6) as the new parent. However the solution may enlarge the depth of the multicast tree and the grandfather of *H*7 is *H*3 which may not be helpful for increasing *H*3's bandwidth.

We prefer to the node with maximum bandwidth rank whenever we choose the removed node or the new parent node. The same reason is that they have enough bandwidth to process tree adjustment algorithm.

The algorithm of bandwidth monitor and self-adaptive tree adjustment algorithm are depicted as follows:

All messages are depicted by the message type (such as UPDATE) and the destination node object. The destination node object has two attributes: address and rank.

```
Algorithm: bandwidth_monitor()
Do while true
  bw = probe_bw();
  rank = int(bw/Br);
  node = getNode(FT,I);
  If (rank!= node.rank)
    update_FT_rank(msg.node);
    UPDATE_msg = msg_constructor(UPDATE,node);
    send(getNode(FT,P).address,UPDATE_msg);
  Endif
  If (rank == 0)
     tree_adjustment();
  Endif
  wait_for_next_probe(INTERVAL);
Enddo
```

All messages are depicted by the message type (such as UPDATE) and the destination node object. The destination node object has two attributes: address and rank.

```
Algorithm: tree_adjustment()
Do while true
 msg = receive_msg();//receive_msg() will be blocked
till a message is received or timeout
```

```
 If (getType(msg) = = UPDATE)
   update_FT_rank(msg.node);
   If isMyself(msg.node) && msg.node.rank= =0
    ADJUST_msg = msg_constructor(ADJUST,msg.node);
    send(getNode(FT, P).address, ADJUST_msg);
   Endif
Else If (getType(msg) = = ADJUST)
   parent_node = maxRank(FT,C);
   PARENT_msg = msg_constructor(PARENT,parent_node);
   send(msg.node.address, PARENT_msg);
Else If (getType(msg) = = PARENT)
   remove_node = maxRank(FT,C);
   ADD_msg = msg_constructor(ADD,remove_node);
   send(msg.node.address, ADD_msg);
Else If (getType(msg) = = ADD)
   add_FT_child(msg.node);
   provide_service(msg.node.address);
   ADD_OK_msg = msg_constructor(ADD_OK, msg.node);
   send(getSourceAddr(msg), ADD_OK_msg);
 Else If (getType(msg) = = ADD_OK)
     remove_FT_child(msg.node);
     stop_service(msg.node.address);
 Endif
Enddo
```

The disadvantage of our algorithm is that it will cost more local resources to process the bandwidth probe and move children transparently. Through analyzing the algorithms of the *bandwidth_monitor*() and the *tree_adjustment*(), we find the probe interval is the main factor which influence the host's performance. The shorter interval can probe bandwidth more exactly while it will cost a lot. We can adjust the probe interval dynamically based on the bandwidth rank to balance the profit and the cost. A simple mechanism is described as follows:

$$probe\_interval_{i+1} = (rank_i + 1) * \text{INTERVAL} \quad (4)$$

The constant INTERVAL is the minimum probe interval. If the bandwidth rank equals to zero, the probe interval will be the INTERVAL. The probe interval can increase with the available bandwidth rank increase.

4 Evaluation

In this section, we present simulation results to evaluate the performance of the proposed self-adaptive tree adjustment mechanism.

4.1 Simulation Environment

We generate *Transit-Stub* network topology with 1000 nodes using GT-ITM [14]. On top of the physical network, multicast group members are attached to stub nodes. The

multicast group size ranges from 100 to 1000 members. The transit-to-transit, transit-to-stub, and stub-to-stub link bandwidth are assigned between 10Mbps and 100 Mbps. The links from edge routers to end systems have bandwidth between 1Mbps and 10Mbps. In simulation environment, the speed of transmitting data in a certain application is often invariable. However, many factor such as congestion or new application's join will change the available bandwidth in the real networks. Here, we choose some members randomly to change there available bandwidth through another program running on these members. The program can occupy the different bandwidth in the different time. The percentage of these members with variable bandwidth is defined as *bandwidth change rate*. We can control the percentage. The minimum required bandwidth of the multicast application is set to 384KB.

4.2 Performance Metrics

Our proposed SAB mechanism can be added to application-layer multicast protocols to avoid node overload. We choose TAG [10,11] as the basic protocol in the experiment. TAG is more attractive in most ALM protocols because it takes the underlying network topology data and the bandwidth into account when constructing the multicast overlay network. The mean *Relative Delay Penalty* and *Link Stress* are same as TAG for the reason that we just add the bandwidth monitor and the tree adjustment algorithms to TAG. So we will not evaluate these two performance metrics which can be found in [10,11]. The performance metrics used for our experiments are the *Member Disconnection Latency* (*MDL*) and the *overhead*.

The *MDL* denotes how long the member suffers disconnection in the multicast service. The latency is longer and the quality of the service will be lower. We report the mean of the latency of all *n* members.

$$Mean\ MDL = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(Ta_{ij} - Td_{ij}) \tag{5}$$

In equation (5), *Ta* is the time when the node acquire the service again from disconnection and *Td* is the disconnection start time. *m* denotes how many times the disconnection occurs.

The overhead is defined as the number of control messages processed by multicast members for maintain the overlay network. We also use the mean of all nodes' overheads to evaluate the performance.

4.3 Results and Analysis

We will evaluate the mean *MDL* and overhead in two different conditions. We fix the value of the INTERVAL (the minimum of the probe interval) with 500 ms in the experiments.

We attach 500 multicast group members to stub nodes randomly and adjust the bandwidth change rate from 10% to 80% in the first experiment. The relationship between the performance metrics and the bandwidth change rate is shown in Fig. 2.

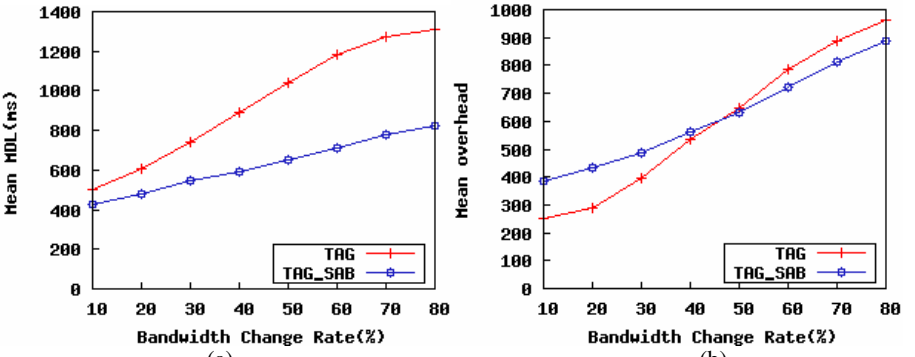


Fig. 2. Performance metrics versus bandwidth change rate

In Fig. 2, TAG denotes the original TAG protocol and TAG_SAB denotes TAG with our SAB mechanism.

From the experimental data showed in Fig. 2(a), we find the mean *MDL* of two mechanisms become higher when the bandwidth change rate increases. However, the mean *MDL* of our mechanism is much lower than that of TAG. The reason is that our mechanism avoids a lot of disconnections by probing available bandwidth before the bandwidth uses up. Just when there are not any nodes can be the new parent of a node with zero bandwidth rank, the disconnection will occur. A node of TAG may lose all its children if the node faces a heavy bandwidth burden. So these children have to send join message again to acquire the multicast service. It will lead mean disconnection latency increase.

Just as we see through Fig. 2(b), the overhead of our TAG-SAB mechanism is higher than that of TAG when the bandwidth change rate is not very high. The reason is that the bandwidth monitor of the SAB mechanism must probe the available bandwidth periodically. However, the situation changes when the bandwidth change rate reaches to 50%. A node in TAG who wants to rejoin the multicast tree needs to send JOIN message to the source node. The source node must probe the *spath* (the

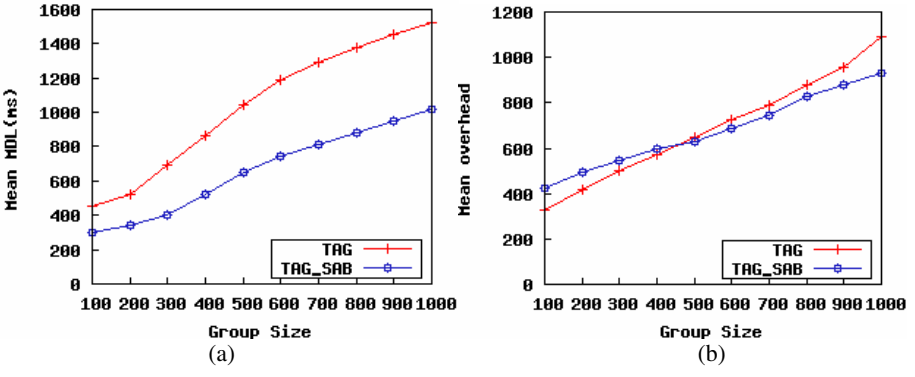


Fig. 3. Performance metrics versus group size

spath of a node refers to a sequence of routers comprising the shortest path from the source node to this node according to the underlying routing protocol) to the new member and execute the path matching algorithm in some nodes. These two steps will cost a lot.

In the second experiment, we change the group size from 100 to 1000 and fix the bandwidth change rate with 50%. The result is shown in Fig. 3.

We can get the similar results as the first experiment from Fig. 3. Our self-adaptive adjustment algorithm does a good job even when the group size grows to 1000.

It can be concluded from the simulation results that our algorithm can decrease mean member disconnection latency and add no more overhead than TAG when the bandwidth changes frequently or the group size is large.

5 Conclusion and Future Work

In this paper we have proposed a self-adaptive bandwidth adjustment mechanism of application-layer multicast. We add bandwidth monitor and tree adjustment algorithms to ALM protocols to automatically adjust the number of children nodes. Bandwidth rank is used to measure the available bandwidth ability of the node. The results show that our mechanism can get higher performance than TAG in the unstable environment. In fact, our mechanism can add to other application-layer multicast protocols too. To saving the host resources which is used to process our mechanism, we bind the probe interval with the bandwidth rank.

For simplicity, the topology of simulation environment is not complex and sweeping enough. Future work will be done on testing and optimizing our self-adaptive adjustment mechanism in a more complex simulation environment and real networks.

References

1. Kosiur, D., Multicasting, I.P.: The Complete Guide to Interactive Corporate Networks. John Wiley & Sons, Inc., Chichester (1998)
2. Chu, Y., et al.: A Case for End System Multicast. IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast (2002)
3. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R., Kubiatowicz, J.: Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In: NOSSDAV 2001. Eleventh International Workshop on Network and Operating Systems Support for Digital Audio and Video (2001)
4. Chawathe, Y.: Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service. Ph.D. Thesis, University of California, Berkeley (December 2000)
5. Chu, Y.-H., Rao, S.G., Zhang., H.: A Case for End System Multicast. In: Proceedings of ACM SIGMETRICS (June 2000)
6. Jannotti, J., Gifford, D., Johnson, K., Kaashoek, M., O'Toole, J.: Overcast: Reliable Multicasting with an Overlay Network. In: Proceedings of the 4th Symposium on Operating Systems Design and Implementation (October 2000)

7. Pendarakis, D., Shi, S., Verma, D., Waldvogel, M.: ALMI: An Application Level Multicast Infrastructure. In: Proceedings of 3rd Usenix Symposium on Internet Technologies & Systems (March 2001)
8. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level multicast using content-addressable networks. In: Proceedings of 3rd International Workshop on Networked Group Communication (November 2001)
9. Jin, X., Wang, Y., Chan, S.-H.G.: Fast overlay tree based on efficient end-to-end measurements. In: Proceedings of IEEE International Conference on Communications (ICC), Korea, 16-20 May, 2005 (2005)
10. Kwon, M., Fahmy, S.: Path-aware overlay multicast. Computer Networks: The International Journal of Computer and Telecommunications Networking 47(1), 23–45 (2005)
11. Kwon, M., Fahmy, S.: Topology-Aware Overlay Networks for Group Communication. In: Proc. of ACM NOSSDAV, pp. 127–136. ACM Press, New York (May 2002)
12. Yeo, C.K., Lee, B.S., Er, M.H.: A Survey of application level multicast techniques. Computer Communications, 1547–1568 (2004)
13. Zhong, Y., et al.: Measurement of the effectiveness of application-layer multicasting. In: Proceedings of Instrumentation and Measurement Technology Conference (IMTC), Ottawa, Canada (May 17-19, 2005)
14. Doar, M.: A better model for generating test networks. In: GLOBECOM 1996. Proceedings of IEEE Global Telecommunications Conference, London, UK, pp. 83–96. IEEE Computer Society Press, Los Alamitos (1996)
15. Subramanian, L., Stoica, I., Balalalshnan, H., Katz, R.H.: OverQoS: Offering QoS using overlays. In: 1st HorNefs Workshop (October 2002)

Overlay Multicast Routing Algorithm with Delay and Delay Variation Constraints*

Longxin Lin, Jie Zhou, and Zhao Ye

School of Computer Science and Engineering, South China University Of Technology, Guangzhou 510641, China
{lxlin,jiezhou,zhye}@scut.edu.cn

Abstract. Overlay multicast is considered as a very effective technique to provide large-scale data dissemination over Internet. In overlay multicast, some applications require the messages should arrive at the destinations within a specified delay bound and these applications also require that the destinations receive the messages from source at approximately the same time. It is defined as the problem of Delay and delay Variation Bounded Multicasting Network (DVBMN) and has been proved to be an NP complete problem. In this paper, by improving the CHAINS algorithm, we present a more efficient heuristic algorithm FCHAINS. We have proved the correctness of our algorithm theoretically, and shown that our algorithm is obviously better than CHAINS in terms of time complexity by performance experiments.

Keywords: Multicast, Overlay Network, Delay and Delay Variation.

1 Introduction

Overlay multicast is considered as a very effective technique to provide large-scale data dissemination over Internet. In overlay multicast, certain nodes form a virtual network (called overlay network), and multicast delivery structures are constructed on top of this virtual network. The existed overlay multicast solutions can be divided into two categories: peer-to-peer architecture and proxy-based architecture. In peer-to-peer architecture, group members (usually end-hosts) are organized to replicate and forward packets to each other. There are many typical examples of peer-to-peer architecture, such as Bayeux [1], Scribe [2], SplitStream [3] and CoolStreaming [4]. In proxy-based architecture, networking service is provided through a set of special nodes (usually dedicated servers) called Multicast Service Nodes (MSN). The MSNs communicate with end-hosts and with each other using standard unicast mechanism. OMNI [5], Overcast [6] and Scattercast [7] belong to this architecture.

* This research is supported by the National Basic Research Program of China (2003CB314805), the National Facilities and Information Infrastructure for Science and Technology Program of China (2005DKA64001), and the 2005 Guangdong-Hong Kong Technology Cooperation Funding Scheme of China: "IPv6 Core Router Research and Development".

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 372–381, 2007.

Some group communication applications such as video conferences, online games and interaction simulations require the messages should arrive at the destinations within a specified delay bound. Furthermore, they also require that the destinations receive the messages from source at approximately the same time. The requirements of these applications can be defined as the problem of Delay and delay Variation Bounded Multicasting Network (DVBMN). It is to find a subnetwork given a source and a set of destinations that satisfies the constraints on the maximum delay from the source to any of the destinations and on the maximum inter-destination delay variation. The problem has been proved as an NP complete problem [8], and some heuristics have proposed such as DVBMA [8], DDVCA [9] and CHAINS [10]. Among these heuristics, CHAINS has the best time complexity of $O(|E|+nk\log(|E|/n)+m^2k)$.

In this paper, we present a new heuristic called FCHAINS (Faster CHAINS). It improves the CHAINS algorithm further, and the time complexity is $O(|E|+nk\log(|E|/n)+mk)$. The main contribution of CHAINS is using a novel method with a time complexity of $O(m^2k)$ to choose a path for each of the destinations from a set of k shortest paths (to each destination) such that the delay variation is the smallest. We present another more effective method with a time complexity of $O(mk)$ to finish the same task. Because in a real overlay network, compared with k, m can be very large (even equals to n), so the improvement has the certain value. We have proved the correctness of our heuristic algorithm solidly in theory, implemented and compared FCHAINS with CHAINS by experiments, and the results show that our heuristic algorithm is much better than CHAINS in terms of computation performance.

This paper is organized as follows: The network model and formal definition of DVBMN is given in section 2. In section 3, we describe CHAINS concept briefly. The detailed explanation of our heuristic algorithm, its correctness and time complexity are discussed at section 4. In section 5, performance studies of CHAINS and FCHAINS are presented, and we conclude in section 6.

2 Network Model and Problem Definition

The overlay network can be modeled by a weighted graph $G=(V,E)$ where V is a set of vertices representing end-hosts (MSNs, in case of proxy-based overlay), and E is a set of edges representing unicast paths. We use the terms “host” or “node” to refer to the vertices and “link” to refer to edges in the overlay graph. For each link in E, define a link-delay function $D: E \to R^+$. The link-delay function is associated a delay with each link in the overlay network. We use the term “multicast” to refer to the task of distributing a message from a source host to a subset of hosts in the overlay network. In overlay multicast, a source host $s \in V$ sends messages to a group of destination hosts $M \subseteq V - s$. The messages are transmitted through a subnetwork $T=(V_T,E_T)$, where T spans the source node s and all the destination nodes in M. The subnetwork T may contain nodes other than M and the source node s. A path $P_T(s,v)$ is defined as the path from source s to $v \in M$ in T. The total delay of

sending a message from s to v along the path will be $\sum_{l \in P_T(s,v)} D(l)$. Define two parameters for DVBMN problem:

1. Source-destination delay bound Δ: the parameter refers to the upper bound on the end-to-end delay along any path from the source to a destination node.
2. Delay variation tolerance δ: the parameter is the maximum allowed difference between the end-to-end delays along the paths from the source to any two destination nodes.

The formal definition about DVBMN is stated below [10].

Given an overlay network $G=(V,E)$, a source node $s \in V$, a multicast group $M \subseteq V-s$, a link-delay function $D: E \to R^+$, a delay bound Δ, and a delay variation tolerance δ, to find a multicast subnetwork $T=(V_T,E_T)$, which spans s and all the nodes in M such that:

$$\sum_{l \in P_T(s,v)} D(l) \le \Delta \text{ for each } v \in M \tag{1}$$

$$\left| \sum_{l \in P_T(s,v)} D(l) - \sum_{l \in P_T(s,u)} D(l) \right| \le \delta, \ \forall v,u \in M \tag{2}$$

Rouskas and Baldine have proved the DVBMN problem is NP-complete and presented the first heuristic DVBMA [8], the complexity of the heuristic is $O(k^2mn^4)$, where k is the number of shortest paths determined between source and destination nodes, m is the number of destination nodes, and n is the number of nodes in the overlay network. Recently, Banik etc. present CHAINS algorithm [10]. The time complexity of CHAINS is $O(|E|+nk\log(|E|/n)+m^2k)$ using the best known k shortest path algorithm [11], where $|E|$ is the number of edges in the overlay network.

3 CHAINS Concept

CHAINS defines a special parameter δ_T, as follows:

$$\delta_T = \max \left| \sum_{l \in P_T(s,v)} D(l) - \sum_{l \in P_T(s,u)} D(l) \right|, \quad \forall v,u \in M \tag{3}$$

It also provides a conception called "tightest delay variation", which is the one that minimizes δ_T .CHAINS algorithm works as follows:

- It first computes the k shortest paths from source to each of the destinations such that the delay of each shortest path is less than or equal to the delay bound Δ. The time complexity of this stage is $O(|E|+nk\log(|E|/n))$.
- Secondly, it selects a shortest path for each destination node from the k shortest paths available in such a way that the delay variation is the smallest possible. Consider the overlay network given in Fig.1 [10]. V_s is the source node and V_2,

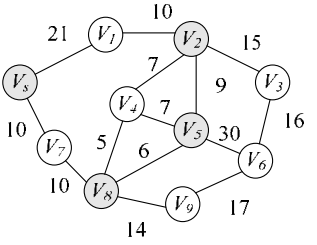


Fig. 1. An example of an overlay network with link delays

Table 1. The list of paths from V_s to V_2, V_5 and V_8 and their corresponding end-to-end delays

source	destination	path	delay
V_s	V_2	(a) V_s - V_1 - V_2	31
		(b) V_s - V_7 - V_8 - V_4 - V_2	32
		(c) V_s - V_7 - V_8 - V_5 - V_2	35
		(d) V_s - V_7 - V_8 - V_5 - V_4 - V_2	40
	V_5	(e) V_s - V_7 - V_8 - V_5	26
		(f) V_s - V_7 - V_8 - V_4 - V_5	32
		(g) V_s - V_1 - V_2 - V_5	40
		(h) V_s - V_1 - V_2 - V_4 - V_5	45
	V_8	(i) V_s - V_7 - V_8	20
		(j) V_s - V_1 - V_2 - V_4 - V_8	43
		(k) V_s - V_1 - V_2 - V_5 - V_8	46

V_5 and V_8 are the destination nodes, and Let $\Delta = 50$. First, CHAINS find all the paths from V_s to V_2, V_5 and V_8 for which the delays are less than or equal to 50. The paths are shown in Table.1 [10].

CHAINS will choose the paths (d), (g) and (j) and merge these paths to be a subnetwork as the solution of DVBMN. The main innovation of CHAINS is to present an algorithm that has a time complexity of $O(m^2k^2)$ to choose a path for each of the destinations from a set of k shortest paths such that the delay variation is the smallest. The key thoughts will be described as follows:

- Suppose source s wants to multicast messages to destination nodes $v_1, v_2, ..., v_m$ and let there be k different shortest paths for each v_i. Let the end-to-end delays of these paths in the nondecreasing order as $d_{i_1}, d_{i_2}, ..., d_{i_k}$.
- Define $S_i = \{d_{i_1}, d_{i_2}, ..., d_{i_k}\}$, $1 \le i \le m$ and let $D = \bigcup_{i=1}^{m} S_i = \{d_1, d_2, ..., d_{mk}\}$, where the elements in D are nondecreasing. For the ith element d_i in D, define $color(d_i) = j$, if $d_i \in S_j$.

– Construct an array $next$ of size mk, where $next[i]$ corresponds to the ith element of the set D. $next[i] = \min\{j \mid j > i, color(d_j) \neq color(d_i), i, j \leq mk\}$, if there exists such a j, otherwise, $next[i] = -1$. In Fig.1, for destination nodes V_2, V_5 and V_8, the corresponding sets are $S_1 = \{31,32,35,40\}$, $S_2 = \{26,32,40,45\}$ and $S_3 = \{20,43,46\}$. The data table is shown in Table 2.

Table 2. The data Table of CHAINS

	1	2	3	4	5	6	7	8	9	10	11
D	20	26	31	32	32	35	40	40	43	45	46
color	3	2	1	1	2	1	2	1	3	2	3
next	2	3	5	5	6	7	8	9	10	11	-1

– The sequence $i, next(i), next(next(i)), next(next(next(i))), \ldots$ is defined as a *chain* starting from the ith element of D. A chain is *valid* when it contains exactly m elements and each element in the chain is of a different color. The value of a valid chain is defined as the difference between the last and first element of the chain.
– The valid chain whose value is minimum among all the valid chains is the solution of the problem. CHAINS presents a algorithm with the time complexity of $O(m^2k^2)$ to find the valid chain with minimal value. The authors also provide an improved algorithm with the time complexity of $O(m^2k)$.

Generally, in a real overlay network, m may be very large and k is very small. So, compared with the $O(m^2k^2)$ algorithm, the $O(m^2k)$ improved algorithm is not of much value in terms of time complexity. In this paper, we give a novel algorithm to find the valid chain with minimal value among all the valid chains. It has $O(mk)$ time complexity, and we name it FCHAINS (Faster CHAINS). In the next section, we will describe it in detail.

4 FCHAINS Algorithm

The main difference between FCHAINS and CHAINS is using a different approach to find the chain with minimal value among all the valid chains. FCHAINS don't use $next$ array. Table 3 is an example of FCHAINS data table for Fig.1.

We define a term "*feasible chain*" as follows, which is similar to the term "*valid chain*" of CHAINS. In this section, we use some of conceptions defined at section 3.

Table 3. The data table of FCHAINS

	1	2	3	4	5	6	7	8	9	10	11
D	20	26	31	32	32	35	40	40	43	45	46
color	3	2	1	1	2	1	2	1	3	2	3

Definition 1. A sequence $c=\{d_{i_1},d_{i_2},...,d_{i_m}\}$ with m different elements of D, where $d_{i_1}\le d_{i_2}\le...\le d_{i_m}$ and $color(d_{i_j})\ne color(d_{i_k})$ if $j,k=1,2,...,m$ and $j\ne k$, is called a *feasible chain*. For a feasible chain c, define the first element of c as $first(c)$ and the last element as $last(c)$.The *value* of a *feasible chain* c, $value(c)$, is defined as the difference between the $last(c)$ and the $first(c)$.

According to definition 1, each feasible chain can be a candidate solution of the problem. Let S_c is the set of all the feasible chains. The objective is to find the feasible chain in S_c with the minimum value, and we define such a feasible chain as $c_{\min}$, that is $value(c_{\min})=\min_{c\in S_c} value(c)$.

Definition 2. For each element $d_i\in D, 1\le i\le mk$, define $C_i=\{c\,|\,first(c)\ \ is\ \ d_i, c\in S_c\}$ and we call C_i the set of feasible chains starting from d_i (notes: for some elements of D, maybe $C_i=\varnothing$). The following lemmas are clear:

Lemma 1. $S_c=\bigcup_{i=1}^{mk}C_i$. □

Lemma 2. For $1\le i\le mk$, if the feasible chain $c_{i_{\min}}\in C_i$ and $value(c_{i_{\min}})=\min\{value(c)\,|\,c\in C_i\}$, then $value(c_{\min})=\min_{1\le i\le mk} value(c_{i_{\min}})$. □

Definition 3. For each element $d_i\in D, 1\le i\le mk$, suppose there is a feasible chain $c_i\in C_i$, if $last(c_i)=d_j$ and $j=\min_{c\in C_i}\{k\,|\,last(c)=d_k\}$, then we call c_i the first feasible chain starting from d_i.

Theorem 1. For each $d_i\in D, 1\le i\le mk$, if c_i is the first feasible chain starting from d_i then $value(c_i)=value(c_{i_{\min}})$.

Proof. By the definition of c_i and $c_{i_{\min}}$, the result is clear. □

Definition 4. For each element $d_i\in D, 1\le i\le mk$, define $C_i'=\{c\,|\,last(c)\ is\ d_i, c\in S_c\}$ and call C_i' the set of feasible chains ending at d_i (notes: for some elements of D, maybe $C_i'=\varnothing$). Clearly, $S_c=\bigcup_{i=1}^{mk}C_i'$.

Definition 5. For each element $d_i\in D, 1\le i\le mk$, suppose the feasible chain $c_i'\in C_i'$, if $first(c_i')=d_j$ and $j=\max_{c\in C_i}\{l\,|\,first(c)=d_l\}$, then c_i' is called the last feasible chain ending at d_i.

Theorem 2. For each $d_i\in D, 1\le i\le mk$, if there is a first feasible chain starting from d_i, say c_i, and let $last(c_i)=d_j$, then there must exist a last feasible chain c_j' ending at d_j. Further assume $first(c_j')=d_l$, then $value(c_j')=\min_{i\le k\le l}\{value(c_{k_{\min}})\}=\min_{i\le k\le l}\{value(c_k)\,|\,c_k\in C_k\}$ and $color(d_l)\ne color(d_{l+1})$. That is, c_j' has the minimal value among all the feasible chains of $S=\bigcup_{k=i}^{l}C_k$ and the first two elements of c_j' must have different colors.

Proof. We first prove the existence of $c_j^{'}$. By the assumption of the existence of c_i and $last(c_i) = d_j$, c_i is also a feasible chain ending at d_j, so $C_j^{'} \neq \varnothing$. By definition 4 and 5, there must exist a last feasible chain $c_j^{'}$ ending at d_j.

By the assumption of $first(c_j^{'}) = d_l$, clearly, for each first feasible chain $c_k \in C_k$, $i \le k \le l$, $last(c_k) = d_j$. So, according to $last(c_j^{'}) = d_j$ and theorem 1, it follows $value(c_j^{'}) = \min_{i \le k \le l}\{value(c_{k_{\min}})\} = \min_{i \le k \le l}\{value(c_k) | c_k \in C_k\}$. If $color(d_l) = color(d_{l+1})$, according to the definition 5, it follows $first(c_j^{'}) = d_{l+1}$, however, it contradicts the assumption of $first(c_j^{'}) = d_l$, therefore $color(d_l) \neq color(d_{l+1})$. □

By theorem 2, let the first feasible chain starting from d_1 be c_1 and $last(c_1) = d_{j_1}$, and the last feasible chain ending at d_{j_1} is $c_{j_1}^{'}$, let $first(c_{j_1}^{'}) = d_{l_1}$, we denote this process as $(d_1, c_1, d_{j_1}, c_{j_1}^{'}, d_{l_1})$; Then, let the first feasible chain starting from d_{l_1+1} be c_{l_1+1} and $last(c_{l_1+1}) = d_{j_2}$, and the last feasible chain ending at d_{j_2} is $c_{j_2}^{'}$, let $first(c_{j_2}^{'}) = d_{l_2}$, we denote this process as $(d_{l_1+1}, c_{l_1+1}, d_{j_2}, c_{j_2}^{'}, d_{l_2})$. Continue to the similar process, we can get a sequence $(d_1, c_1, d_{j_1}, c_{j_1}^{'}, d_{l_1})$, $(d_{l_1+1}, c_{l_1+1}, d_{j_2}, c_{j_2}^{'}, d_{l_2})$,…, $(d_{l_{k-1}+1}, c_{l_{k-1}+1}, d_{j_k}, c_{j_k}^{'}, d_{l_k})$,…, $(d_{l_{N-1}+1}, c_{l_{N-1}+1}, d_{j_N}, c_{j_N}^{'}, d_{l_N})$.

Theorem 3. $value(c_{\min}) = \min\{value(c_{j_k}^{'}) | 1 \le k \le N\}$.

Proof. According to theorem 2, for each k, $1 \le k \le N$, $value(c_{j_k}^{'}) = \min_{l_{k-1}+1 \le i \le l_k}\{value(c_i) | c_i \in C_i\}$. By the definition of $(d_{l_{k-1}+1}, c_{l_{k-1}+1}, d_{j_k}, c_{j_k}^{'}, d_{l_k})$, $1 \le k \le N$, we can obtain $S_c = \bigcup_{k=1}^{N} \bigcup_{i=l_{k-1}+1}^{l_k} C_i$. Lemma 2 implies that $value(c_{\min}) = \min\{value(c_{j_k}^{'}) | 1 \le k \le N\}$. □

Our **FCHAINS algorithm** is based on Theorem 3. It works as follows:

1. Set $value(c_{\min}) = \infty$ .The loop variable j traverses the array D from the first element to the last.
2. When we find out the first feasible chain c_1 with $last(c_1) = d_{j_1}$, we also find out the last feasible chain $c_{j_1}^{'}$ ending at d_{j_1} and $first(c_{j_1}^{'})$ (that is, d_{l_1}). We will save $c_{j_1}^{'}$ to a double linked list.
3. Compare $value(c_{j_1}^{'})$ with $value(c_{\min})$, and assign the smaller to $value(c_{\min})$.
4. Delete the first element d_{l_1} of $c_{j_1}^{'}$ from the double linked list, increment j by one, and continue to traverse the array D from d_{j_1+1}.
5. Loop the steps 2, 3 and 4 until j reaches to the last element of D.
6. $c_{\min}$ is the solution.

For example, using the data given in Table 3, the computation process is shown as Table 4.

Table 4. The computation process of FCHAINS, N/A = Not Available

k	start element($d_{l_{k-1}+1}$)	$c_{l_{k-1}+1}$	c'_{j_k}	$c_{\min}$
1	d_1	d_1 - d_2 - d_3	d_1 - d_2 - d_3	d_1 - d_2 - d_3
2	d_2	d_2 - d_3 - d_9	d_7 - d_8 - d_9	d_7 - d_8 - d_9
3	d_8	d_8 - d_9 - d_{10}	d_8 - d_9 - d_{10}	d_7 - d_8 - d_9
4	d_9	N/A	N/A	d_7 - d_8 - d_9

Then, let's analyze the algorithm's time complexity. Because the algorithm has only one loop and the maximum iteration number is mk , so its time complexity is $O(mk)$.

The whole FCHAINS algorithm can be divided into three stages: (1) computes the k shortest paths [11] for each destination node as in CHAINS, the time complexity is $O(|E|+nk\log(|E|/n))$. (2) Constructs the FCHAINS data table according to the known mk shortest paths. The time complexity of this step is $O(mk)$. (3) Find out the feasible chain with minimal value using the FCHAINS data table, its time complexity is $O(mk)$. So, the whole time complexity of FCHAINS algorithm is $O(|E|+nk\log(|E|/n)+mk)$.

In real network environment, usually k is very small, but n and m may be very large, so reducing the time complexity from $O(m^2k)$ to $O(mk)$ is valuable.

5 Performance Evaluation

For evaluation purposes, we have implemented CHAINS and FCHAINS using C++, and designed two experiments.

Experiment 1. Performance comparison of finding out the feasible chain with minimal value. Suppose we have gotten each destination node's k shortest paths from source s according to k shortest path algorithm and constructed the FCHAINS and CHAINS data table respectively. We use the random numbers to represent the actual path delay, and let $\Delta=1000$. Then, we evaluate the algorithms' performance by adjusting the size of k and m . In order to obtain the random numbers with more uniform distribution, we have gathered 120,000 random integers with the range of [10, 1000] from website [13] and taken it as random number sources. The experiment has be done on a personal computer (2.6 GHz C4 Intel CPU, 512 MB RAM, Linux Red Hat 9.0). The test results are shown in Table 5.

From Table 5, we observe that FCHAINS outperforms CHAINS in terms of execution time too much. We also observe that the execution time of CHAINS increases very quickly with m and k growing, and when m becomes more and more large, compared with CHAINS, the execution time of FCHAINS is so small that we can ignore it.

Table 5. The results of experiment 1(C: CHAINS, FC: FCHAINS, Running Time: ms)

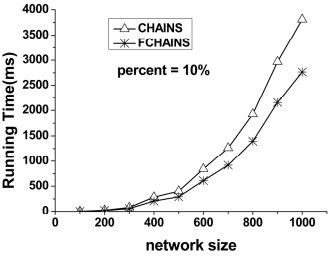
	k =2		k =4		k =6		k =8	
m	C	FC	C	FC	C	FC	C	FC
100	0.7151	0.027	2.0158	0.052	3.537	0.0778	5.2546	0.1033
500	17.553	0.133	50.681	0.261	91.382	0.3905	137.55	0.5210
1000	70.832	0.268	204.66	0.582	375.50	0.8087	563.87	1.0633
3000	637.66	0.855	1853.8	1.731	3460.4	2.4632	5131.03	3.2586
5000	1797.9	1.437	5169.1	3.116	9682.1	4.4238	14652.3	5.5335
8000	4557.1	3.024	13393	5.865	25014	8.1937	37852.9	11.076
10000	7118.9	3.548	20990	7.290	39201	12.546	59383.2	15.022

Experiment 2. To compare the whole execution time, we have run FCHAINS and CHAINS algorithms on random graphs constructed by Georgia Tech Internet Topology Models (GT-ITM) [12]. The nodes in graphs are placed in a grid of dimension 3100 x 3100 km and the delay for each link is set to the propagation delay of electrical signal along the link, we can compute link's delay as the equation (4).

$$d_{uv} = \frac{Length(u,v)}{c} \tag{4}$$

Where c is the speed of light, $Length(u,v)$ can be obtained from the output files of GT-ITM. We set Δ=0.05s, k=3. For the sake of convenience, we map Δ to link's length according to equation (4), so Δ=1500 km. The average node degree for each graph is kept in the range of 3.5 to 6 through adjusting parameter α. CHAINS and FCHAINS have tested on various graphs with number of nodes varying from 100 to 1000 and the percentage of nodes (p) in the multicast group varying from 10 percent to 30 percent. The experiment's software and hardware platform is the same as experiment 1. The results are plotted in Figs.2 and 3. Each point in the plots represents the average value taken over 10 graphs.

From Fig.2 and 3, we observe that FCHAINS algorithm outperforms CHAINS not so much as experiment 1 and compared with CHAINS, the improvement rate of FCHAINS varies from 20% to 30%. Although the improvement is not multi-times, it is still very effective. The results can be explained easily, because the k shortest paths algorithm needs to consume much execution time.



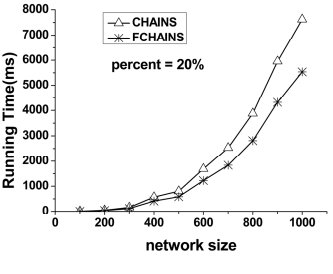


Fig. 2. Execution Time graph for p =10% **Fig. 3.** Execution Time graph for p =20%

6 Conclusions

In this paper, we consider the problem of Delay and delay Variation Bounded Multicasting Network (DVBMN). We discuss the recent heuristic algorithm CHAINS which has the best time complexity for the same problem so far, and presented a new improved algorithm FCHAINS, which reduces the time complexity from $O(|E|+nk\log(|E|/n)+m^2k)$ to $O(|E|+nk\log(|E|/n)+mk)$. We prove the correctness of our heuristic algorithm theoretically. We also implement the two heuristic algorithms and compare their performance through experiments, the results show that FCHAINS outperforms CHAINS much more in terms of finding out the feasible chain with minimal value and the total running time. In our network model, we ignore the dynamic nature which may exist in a real overlay network, so, the DVBMN problem may be more complex in fact. As a part of our future work, we will continue to consider the DVBMN problem when link delay is the function of time.

References

1. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiatowicz, J.D.: Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In: pp. 11–20. ACM Press,Port Jefferson, New York, United States (2001)
2. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.I.T.: Scribe: a large-scale and decentralized application-level multicast infrastructure. Selected Areas in Communications, IEEE Journal 20, 1489–1499 (2002)
3. Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., Singh, A.: SplitStream: high-bandwidth multicast in cooperative environments. In: Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM Press, Bolton Landing, NY, USA (2003)
4. Xinyan, Z., Jiangchuan, L., Bo, L., Yum, Y.S.P.: CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. 3, 2102–2111 (2005)
5. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S.: OMNI: An efficient overlay multicast infrastructure for real-time applications, vol. 50. Elsevier Science, Amsterdam (2006)
6. Jannotti, J., Gifford, D., Johnson, K., Kaashoek, F., O'Toole, J.: Overcast: Reliable Multicasting withan Overlay Network, pp. 197–212 (2000)
7. Chawathe, Y.: Scattercast: an adaptable broadcast distribution framework, vol. 9, pp. 104–118. Springer, Heidelberg (2003)
8. Rouskas, G.N., Baldine, I.: Multicast routing with end-to-end delay and delay variation constraints. Selected Areas in Communications, IEEE Journal 15, 346–356 (1997)
9. Sheu, P.R., Chen, S.T.: A fast and efficient heuristic algorithm for the delay-and delay variation-bounded multicast tree problem, vol. 25, pp. 825–833. Elsevier, Amsterdam (2002)
10. Banik, S.M., Radhakrishnan, S., Sekharan, C.N.: Multicast Routing with Delay and Delay Variation Constraints for Collaborative Applications on Overlay Networks. Parallel and Distributed Systems, IEEE Transactions 18, 421–431 (2007)
11. Jimenez, V.M., Marzal, A.: Computing the K shortest paths: a new algorithm and an experimental comparison. In: Vitter, J.S., Zaroliagis, C.D. (eds.) WAE 1999. LNCS, vol. 1668, Springer, Heidelberg (1999)
12. Zegura, E.W., Calvert, K.L., Bhattacharjee, S.: How to model an internetwork. In: INFOCOM 1996, vol. 2, pp. 594–602. IEEE Computer Society Press, Los Alamitos (1996)
13. Random Number Generator Web Site (2007), http://www.random.org/

Selfish MAC Layer Misbehavior Detection Model for the IEEE 802.11-Based Wireless Mesh Networks

Hongjian Li[1], Ming Xu[1], and Yi Li[2]

[1] Department of Network Engineering, School of Computer Science,
National University of Defense Technology, Changsha 410073, China
hongjianli@nudt.edu.cn, xuming-64@hotmail.com
[2] Staff Room of Automatization Technology, Logistics and Engineering University,
Chongqing 400016, China
necklacemary@163.com

Abstract. CSMA/CA, the contention mechanism of the IEEE 802.11 DCF medium access protocol, has recently been found vulnerable to selfish attacks. Such attacks can greatly increase a selfish station's bandwidth share at the expense of honest stations. Based on the in-depth research of the attack model of selfish behavior in WMN, the attack strategy of smart selfish nodes was focused, analyzed and according to its characteristics, a double-mode detection mechanism was proposed. The one mode is selfish attack detection to sequential data; the other is attack detection to non-sequential data transmission ground on statistics. Subsequently the selfish behavior detection model was proposed with the double-mode detection mechanism, which was finally simulated in ns-2. The simulation results indicated that the new selfish behavior detection model which used the double-mode detection mechanism is preferably adapt to the selfish attack behavior and can commendably solve the problem of smart selfish nodes in WMN.

Keywords: selfish behavior, double-mode detection model, IEEE 802.11, Wireless Mesh Networks.

1 Introduction

As a new broadband wireless network structure, Wireless Mesh Network (WMN) has its special characteristics which are different from the traditional wireless network. The obvious advantages of WMN contain deploying network rapidly, improving the coverage ratio, increasing the capacity and reducing initial investment of the network. The technology is especially applied to accessing the broadband wireless to the backbone network. [1]

Operation deviate from legitimate protocol in wireless networks has being received considerable attention from research communities in recent years. The WMN that is a three layer IEEE 802.11-based network with gateway, router and client nodes is

[1] Supported by the National Basic Research Program of China under Grant No. 2006CB303004.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 382–391, 2007.

generally accepted and being familiar with. However, IEEE 802.11 MAC protocol has been designed with the thinking of fully cooperative users, therefore it offers little protection against noncooperative users. Under such circumstances, router and client nodes may not comply with the protocol without being punished. Misbehavior nodes can be divided into two categories, selfish nodes to obtain more resources [5], malicious nodes to destroy the legitimate operation [13][12]. This paper focused on the selfish nodes attack.

The basic IEEE 802.11 MAC layer uses the Distributed Coordination Function (DCF) to share the medium between multiple stations. Thus the selfish nodes will reduce the resources of wireless channel which can be used by the legitimate nodes, thereby affect the network performance, even interrupt the network service. There are two categories of selfish nodes in WMN, selfish client nodes and selfish router nodes. Selfish client nodes access WMN with selfish strategy to achieve greater throughput, reduce power consumption and improve QoS[1]. Selfish router nodes use selfish strategy to result in the congestion of network or even the denial of service. With the characteristics of Multi-hop and public access, it is more vulnerable for WMN to selfish client nodes attack. The selfish attacks in router nodes will also have significantly impact on the entire network performance. Therefore, it is highly necessary to use some appropriate mechanism to detect misbehaviors in WMN.

Based on the detection method of MAC layer selfish behavior in wireless ad hoc networks proposed in [10], this paper presented a selfish behavior detection model with double-mode detection mechanism in WMN. Different detection mechanisms are used for router and client selfish attacks respectively. Finally, the detection model is simulated and subsequently the analysis of the simulation results is given.

2 Related Work

Solutions to MAC layer selfish attacks can be divided into two categories of active and passive ones. The former is correcting the protocol to remove the selfish attack [7][3][2], which is unrealistic in practice. The passive solution is detecting selfish attacks by statistical means, and then making responses to the selfish nodes to reduce the impact of attacks, which is the basis of our work.

Kyasanur and Vaidya [7] have addressed the MAC layer misbehavior using correction mechanism. Their main idea is to let the receiver assign and send back-off values to the sender in CTS and ACK frames and then use them to detect potential misbehavior. The latter is handled using a correction scheme that adds to the next back-off a penalty that is a function of the observed misbehavior. Based on the corrected MAC protocol, the rapid detection method [3] of selfish attacks through access point (AP) is proposed. Although those above solutions can work well in detecting and preventing selfish attacks, it is difficult for real practice as with the existing IEEE 802.11 equipment's compatibility problems.

Early work on MAC layer selfish behavior also includes the discussion about the weaknesses of the 802.11 physical layer and the virtual carrier sense mechanism [6]. For WLAN, trusted agent [8] is described, which detects the real station ID selfish

attacks by statistical method. It shows how a Nash Equilibrium is achieved among selfish users when the cost for accessing the channel repeatedly is being jammed by another node in [9] [11]. References [4] detailedly analysed the selfish attacks in the Ad hoc network. The detection method of selfish attacks in Ad hoc network is presented in [10]. However, those above researches were mostly done in Ad hoc or WLAN. There is no good solution to a long period of testing, particularly for more collaboration nodes and smart selfish node. Work in WMN is rarely at present.

In this paper, we firstly make a deep study on the system and attack detection model of the selfish attacks in WMN, and then we analyzed the characteristics of smart selfish node. Based on the detection method in [10], focused on its incapacity for the collaboration and smart selfish node we proposed a double-mode detection mechanism to solve that weakness, which formed a new selfish behavior detection model used in selfish attack detection in WMN.

3 Attack Detection Model in WMN

According to the features of the WMN and the IEEE 802.11 MAC protocol, we use the following system model of the selfish behavior in WMN and the attack detection model.

3.1 System Model of Selfish Behavior in WMN

1. The IEEE 802.11 WMN (router nodes and client nodes) works in DCF mode, which is the operation mode usually deployed.

As shown in Fig. 1,DCF delays frame transmissions right after the channel is sensed idle for DIFS (DCF InterFrame Spacing) time. It waits for an additional random time, back-off time, after which the frame is transmitted. The back-off time is bounded by the contention window size CW. This is applied to data frames in the basic scheme, and to RTS frames in the RTS/CTS scheme. The back-off time of each station is decreased as long as the channel is idle. When the channel is busy, the back-off time is frozen. When the back-off time reaches zero, the station transmits its frame. If the frame collides with another frame (or RTS), the sender times out waiting for the ACK (or the CTS) and computes a new random back-off time with a larger CW to retransmit the frame with lower collision probability. When a frame is successfully transmitted, the CW is reset to CWmin. The network allocation vector (NAV) of all other stations is set to the frame duration field value in RTS/CTS and DATA headers.

2. To simplify the question, only selfish client under the normal router or selfish router without selfish client within its coverage is considered, as shown in Fig. 2, leaving out of the route-and-client's combinative selfish behavior.

3. Although the router is managed easier than client, the misbehavior may howbeit occur due to the attack of the nodes. Thus selfish behaviors both in the router and the client nodes are considered.

4. The detection system is implemented only on router nodes. No modification or reconfiguration of wireless adapters has to be made on the client side, which is actually unpractical in WMN.

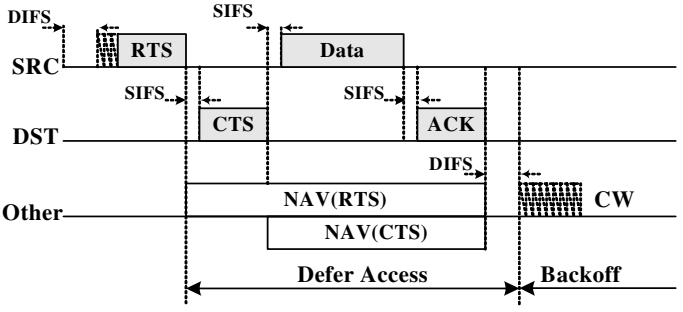


Fig. 1. RTS/CTS/Data/ACK handshaking in DCF mode

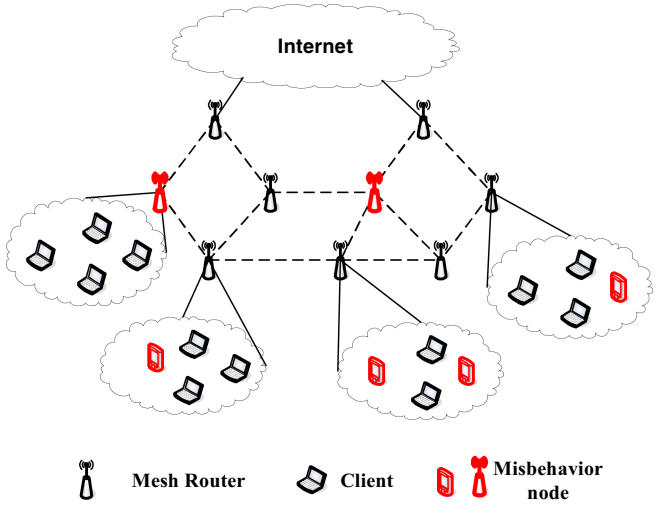


Fig. 2. Architecture of selfish nodes in WMN

3.2 Attack Model and Attack Detection Model

The methods and strategies of the selfish behavior attack in WMN are as follows:

1. Selfish nodes do not comply with rules to set the contention window (CW), no double CW with collision, namely $CW < \min(CW_{\min} * 2^{i}, CW_{\max})$, where i is the number of successive collisions, $CW_{\min}$ is min CW, $CW_{\max}$ is max CW.

2. Back-off values are no longer satisfied with uniform distribution in (0, CW).
3. DIFS, PIFS even SIFS are used improperly.
4. NAV value is set to the larger by RTS/CTS duration field.
5. Smart selfish nodes will carefully use the above four selfish methods, or distantly use the selfish methods to prevent nodes to be found by detection mechanism.

The nodes which are instructed by the MAC protocol to defer transmission are able to overhear transmissions from nodes whose transmission range reside in [10]. Fig. 3 depicts a scenario where node A or B is malicious. At this stage, we assume that A is the only selfish node. Node A accesses the channel by using a randomly selected back-off value within its CW. When the back-off counter decreases to zero, A sends an RTS to B, which replies with a CTS. Node A's RTS message silences nodes 1and 2. Similarly, node B's CTS silences nodes 2 and 3. Following the RTS-CTS handshake, A sends a data segment to B. And the procedure repeats. Consider the *i*-th transmission of node A. A node in its transmission range finds time point t_i of RTS packet reception from

$$t_i = T_{i-1} + T_{SIFS} + T_{ACK} + T_{DIFS} + b_i, i > 1 \tag{1}$$

where T_{i-1} denotes the end time point of reception of the previous data segment and b_i is the random back-off value. Thus, the back-off values can be easily derived.A node within transmission range of B can also compute the back-off value used by using the overheard ACK of the previous data segment transmission. Then, a node can measure time point $t_i^{'}$ of CTS packet reception and compute the back-off value of node A by using

$$t_i^{'} = T_{ACK_{I-1}} + T_{DIFS} + b_i + T_{RTS} + T_{SIFS}, i > 1 \tag{2}$$

where $T_{\mathrm{ACL_{i-1}}}$ denotes the end time point of reception of the previous ACK frame.

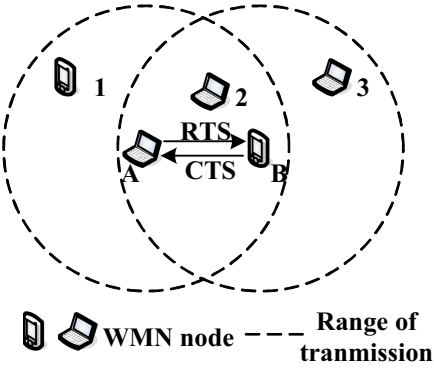


Fig. 3. Observer nodes

4 DOUBLE-MODE Detection Model

●Selfish behavior of router node

Due to some "smart" selfish node can adjust the strategy of selfish behavior according to the detection model in[10], for instance, smart node may stop selfish

behavior before the very moment the detection system can make a judgment. This brought trouble to the sequence judgment method. Therefor we bring forward a double-mode mechanism in the detection model for judgment: (1) the frequency of node's seizing channel behavior during the active time of the node, (2) continuous sampling result of node's back-off value. Continuous sampling k times for selfish node, $b =< b_0, b_1, ..., b_{k-1} >$, where b_i denotes the *i*-th sampling. Detection model M_{ids} is created through training. Calculate the average probability of selfish behavior $(\sum M_{ids}(b_i))/k$, this data is recorded for the Stat. of the seizing channel frequency $\eta_j = (\sum M_{ids}(b_i) + N * \eta_j)/(k+N)$, where j denotes node No., N denotes the sampling time. Compare them separately with continuous sampling threshold and statistical threshold to make the judgment.

● Selfish behavior of client node

As the WMN client node is more difficult to control than router node, besides, it has mobility, the detection model for selfish behavior of client node is as follows:

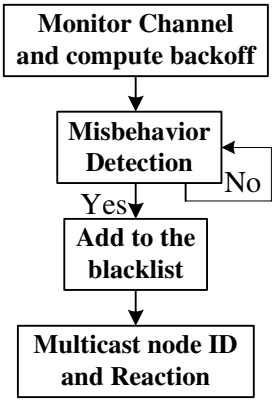


Fig. 4. Detection system

Router nodes sense channel continuously, compute the back-off value, and send the results to the detection module which then use the double-mode detection mechanisms to detect. Once selfish behavior is found, blacklist the node ID which will be also broadcasted to other router nodes to adopt appropriate response mechanism.

● Response Mechanism

It is not enough to blacklist and report to others the node ID, some automatic response mechanism is still required. For selfish behavior of router node, its neighbor node will cut off their communications. For client node, if other client nodes detect its attack behavior, they cut off the communication while making a report to its associated router which will take corresponding measures (Prohibition or selectively prohibit network access).

5 Attack Detection Algorithm in Double-mode Detection Model

Double-mode mechanism attack detection algorithm is as follows

Detection mode 1 (mainly based on [10]): Let the random variable *Y* stand for the back-off value of legitimate node, hence it is uniformly distributed in [0, *CW*]. Also, let the random variable *X* stand for the misbehaving node (attacker), so that it has unknown *f*(*x*) with support [0, *CW*]. Define the probability distribution function set $\psi = \left\{ f(x) : \int_0^{CW} xf(x)dx < CW/2 - \varepsilon \right\}$, where ε is the parameter for the adjustment. If node A is selfish, the probability that the X value is less than Y is $P(X < Y) = \int_0^{CW} P(Y > X \mid X = x) f(x)dx$, and P is larger than 0.5. f_0 and f_1 denote the continuous probability density function, then define results after k times sampling $x_1, \ldots, x_k$ as

$$S_k = \ln \frac{f_1(x_1, \ldots, x_k)}{f_0(x_1, \ldots, x_k)}$$, and $\begin{array}{ll} S_k \geq a \Rightarrow & H_0 \\ S_k < b \Rightarrow & H_1 \\ b \leq S_k < a \Rightarrow & \text{more sampling} \end{array}$, where H_0, H_1 denote the different judgments. The thresholds *a* and *b* are dependent on ε.

Detection mode 2: It is proposed by this paper in allusion to smart selfish node. S_k is processed again. Make statistics of the node's intrusion frequency $R = \int_0^N g(x)dx$, where N denotes the statistical number, $g(x)$ is the *k*-th judgment result. Similarly, we need to give a judgment method based on statistics $\tau = \left\{ g(x) : \int_0^N g(x)dx > E_N(R) - \varepsilon \right\}$, where E denotes the expectation value, ε is a parameter different from the ε in previous sequence detection method. And then $\begin{array}{ll} R_N \geq a_0 \Rightarrow & H_0 \\ R_N < b_0 \Rightarrow & H_1 \\ b_0 \leq R_N < a_0 \Rightarrow & \text{more sampling} \end{array}$, where H_0, H_1 denote the different judgment result. The thresholds are dependent on the choosing of ε.

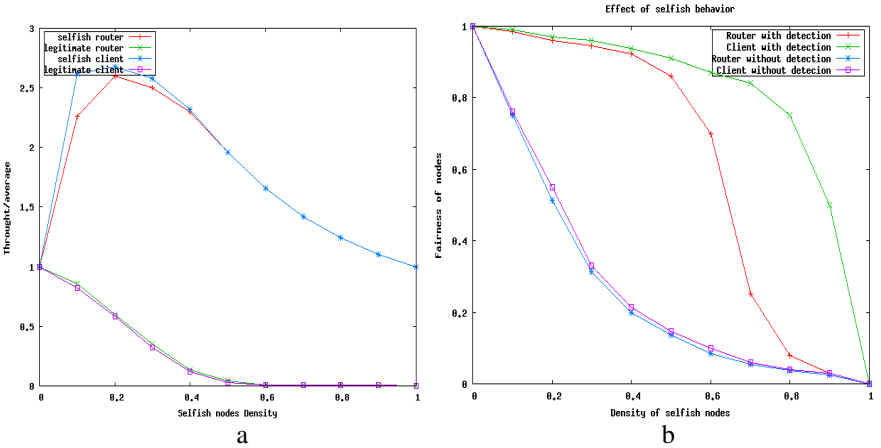
6 Simulation and Discussion

In NS2.29 we achieved the simulation to detect 802.11 MAC selfish behaviors using the double-mode detection model proposed by this paper. Table 1 shows the parameters used in NS2. The adopted wireless channels between the situations when routers communicate with each other and when client access to network are different.

Table 1. NS2 parameters

Parameter	Value
Topology	2000m×2000m
MAC protocol	802.11
Antenna model	OmniAntenna
Router Radio range	300m
Client Radio range	150m
Router bandwidth	11Mbps
Client bandwidth	2Mbps
Packet type	TCP
Route protocol	AODV

Define the fairness of network as $\Sigma S_i / (\Sigma(S_i + L_i))$, where S_i denotes the number of the times of selfish node i's successful MAC layer transmission, L_i denotes the number of the times of legitimate node i's successful MAC layer transmission. The density of selfish nodes is the ratio of the selfish nodes' number to the total nodes' number in network. The density of router nodes and client nodes are calculated separately. Fig. 5 shows that the impact of selfish attack on the network fairness increases rapidly in WMN without our detection model. The other way round, the network fairness can be guaranteed in WMN with our detection model when the density of selfish node is not very high. Comparing with router, the client node in the legitimate router's coverage can prevent the selfish behavior attack better.



a b

Fig. 5. Fairness in WMN

Similarly, the detection rate is defined as the ratio of the number of the selfish nodes that have been detected to the number of the total selfish nodes. Comparing with the sequence method [10], our double-mode detection method's sample time is much less. As shown in fig. 6, our detection model improves the speed of selfish

attack detection as well as ensuring the detection effect of the smart selfish node. When the detection rate is larger than 98%, false detection probability (the ratio of the number of the legitimate nodes that are misjudged to selfish nodes to the number of the legitimate nodes) of router is less than 0.9%, false detection probability of client nodes is less than 0.6%.

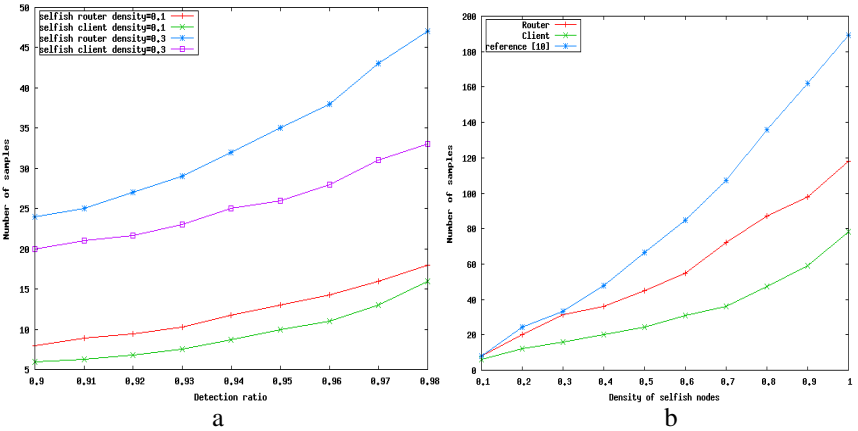


Fig. 6. The number of sampling

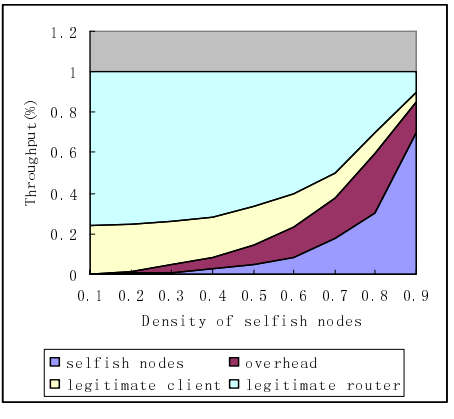


Fig. 7. Network overhead

According to the simulation results (Figure 7), we can see that our double-mode detection model can control the network access of the selfish nodes when the density of selfish node is not very high.

7 Conclusion

In this paper, we deeply analysed the IEEE 802.11 MAC protocol DCF mechanism, the selfish behavior attacks model, selfish attack detection model and smart nodes

selfish attack strategy in WMN. Based on the selfish detection model [10] in Ad hoc network, a double-mode detection mechanism in WMN is proposed, which formed a new attack detection model. Finally the model is simulated in NS-2. Simulation results show that the double-mode detection model is applicable in detecting selfish behavior and smart selfish node in WMN, it can guarantee the network fairness while lower the network overhead.

References

1. Guang, L., Assi, C.: Modeling and Analysis of Predictable Random Backoff in Selfish Environment. In: MSWiM 2006, Terromolinos, Malaga Spain, October 2–6, 2006, pp. 86–90 (2006)
2. Guang, L., Assi, C.: Mitigating smart selfish MAC misbehavior in ad hoc networks. In: Proc. IEEE WiMob (June 2006)
3. Kyasanur, P., Vaidya, N.H.: Detection and Handling of MAC Layer Misbehavior in Wireless Networks. In: Proc. 2003 Int'l Conf. Dependable Systems and Networks, pp. 173–182 (2003)
4. Djenouri, D., Khelladi, L., Badache, A.N.: A Survey of Security Issues in Mobile Ad Hoc and Sensor Networks, Communications Surveys & Tutorials, IEEE, Vol. Communications Surveys & Tutorials 7(4), 2–28 (2005)
5. Kyasanur, P., Vaidya, N.: Selfish MAC layer misbehavior in wireless networks. IEEE Transactions on Mobile Computing (September 2005)
6. Bellardo, J., Savage, S.: 802.11 denial-of-service attacks: real vulnerabilities and practical solutions, In: Proc. USENIX Security Symp., Washington DC, pp. 15–28 (August 2003)
7. Kyasanur, P., Vaidya, N.H.: Detection and handling of MAC layer misbehavior in wireless networks. In: Proc. Int. Conf. Dependable Systems and Networks, San Francisco, CA, pp. 173–182 (June 2003)
8. Raya, M., Hubaux, J.-P., Aad, I.: DOMINO: a system to detect greedy behavior in IEEE 802.11 hotspots. In: Proc. MobiSys, Boston, MA, pp. 84–97 (June 2004)
9. Cagalj, M., Ganeriwal, S., Aad, I., Hubaux, J.-P.: On selfish behavior in CSMA/CA networks. In: Proc. IEEE INFOCOM 2005, Miami, FL, pp. 1514–2513 (March 2005)
10. Radosavac, S., Baras, J.S.: A Framework for MAC Protocol Misbehavior Detection in Wireless Networks. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 33–42. Springer, Heidelberg (2005)
11. Konorski, J.: A Game-Theoretic Study of CSMA/CA Under a Backoff Attack. IEEE/ACM Transactions on Networking 14(6), 1167–1177 (2006)
12. Aad, I., Hubaux, J.P., Knightly, E.W.: Denial of service resilience in ad hoc networks. In: Proceedings of ACM MobiCom (September 2004)
13. Gupta, V., Krishnamurthy, S., Faloutsous, M.: Denial of service attacks at the MAC layer in wireless ad hoc networks. In: Proc. of MILCOM (2002)

rHALB: A New Load-Balanced Routing Algorithm for k-ary n-cube Networks

Huaxi Gu[1], Jie Zhang[2], Kun Wang[3], and Changshan Wang[3]

[1] State key lab of ISN, Xidian University, Xi'an, China 710071
[2] Dept. of Computing and Information Systems, University of Bedfordshire, UK
[3] School of Computer Science, Xidian University, Xi'an, China 710071U
hxgu@xidian.edu.cn, kwang@mail.xidian.edu.cn

Abstract. A new load-balanced routing algorithm for k-ary n-cube networks is proposed in this paper. The new algorithm, called rHALB (r Hop Aware Load-Balanced) routing algorithm, can efficiently balance traffic loads in k-ary n-cube networks. It is based on the limited global information, which is a tradeoff between local and global information. rHALB makes use of the number of potential deadlocked packets to detect the congestion, and can direct the traffic to bypass any detected hotspots, thus balancing the traffic. Simulations are carried out on 2D torus networks by using OPNET. The results show that rHALB achieves a better performance (latency and throughput) under various traffic patterns than dimension order routing and Duato's algorithm, which are popularly used in commercial products, and ROMM, ACQ and GAL, which were proposed in literature recently.

1 Introduction

The direct interconnection networks have been studied extensively as the critical components in many fields, such as multiprocessor systems, I/O interconnect and supercomputers. A survey of such networks can be found in [1,2]. Recently, direct interconnection networks have become popular architectures for on-chip communication. The traditionally used bus-based architecture may not be able to meet the scalability, reliability and high throughput requirement for complex multiprocessor system in future [3]. Hence, the functional IP blocks can communicate with each other with the help of the direct interconnection networks[4]. A k-ary n-cube network is a family of direct interconnection networks. Besides its use in terabit routers, k-ary n-cube networks have been used in supercomputers (or parallel computers) such as SGI Origin 2000, iPSC860, CrayT3D and CrayT3E [1, 2].It has also been used for I/O interconnect in infiniBand architecture [5].

Routing algorithms, which specify how packets travel from the source to the destination node, are crucial for the performance of the networks. An efficient routing algorithm can improve the network throughput and reduce the average latency. Routing algorithms can be generally classified into two categories: deterministic and adaptive. Deterministic routing has been very popular in practice for its simple

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 392 – 401, 2007.

hardware implementation. Dimension order (DO) routing [6] is a typical representative of this kind. It routes the packet first along the lowest dimension and then along higher dimension until the packet arrives at the destination. Many commercial products use dimension order routing, such as Intel Paragon, MIT J-machine and Cray T3D [1, 2]. However, DO cannot avoid congested links, and thus makes some links overloaded while others idle.

Adaptive routing was proposed to overcome the performance limitations of deterministic routing. A minimal adaptive routing algorithm can route packets along any of the shortest path in the topology. Duato proposed an adaptive algorithm in [7]. The algorithm requires at least three virtual channels, which are divided into two classes ***a*** and ***b.*** Class ***b*** contains two virtual channels, in which deterministic routing is applied. The rest virtual channels belong to class ***a***, where fully adaptive routing is used. The packets can adaptively choose any virtual channels available from class ***a***. If all the virtual channels of class ***a*** are busy, the packets enter channels that belong to class ***b***. Duato's algorithm is very popular and has been used in many commercial products [1, 2]. However, it uses DO to provide deadlock freedom, which restricts the use of some virtual channels. Recently, unrestricted true fully adaptive routing (TFAR) has gained consideration in the scientific community [8]-[11]. TFAR does not impose any restriction on routing, so deadlock may occur. TFAR uses deadlock detection mechanisms to find the deadlock and resolve it by the recovery mechanism. Most deadlock detection mechanisms use time-out criteria to ensure deadlock occurrence, which is simple to implement [8, 10, 11].

The routing algorithms mentioned above often focus their attentions on providing low latency on local traffic. But for some adversarial traffic, such as hotspot or tornado traffic [2], these algorithms may load some links heavily while letting others idle. This uneven network utilization often results in an early saturation and hence degrades the network performance. Therefore, some load-balanced routing algorithms are proposed to strike a balance between the conflicting goals of providing low latency on local traffic and providing high throughput on adversarial traffic [12]-[15]. A two-phase randomized routing algorithm is proposed by Valiant in [12]. It uses a random node as intermediate destination so as to give good performance on worst-case scenario. But it lost locality and the buffer size of O(n) are required in each node. ROMM [13] is an improved version of Valiant's algorithm. ROMM chooses the intermediate node within the minimal quadrant. The packet routes along the minimal path, with a randomized order of dimension traversal, from the source to the intermediate node, and repeats the same algorithm from intermediate node to destination. Arjun [14] proposes a load-balanced routing algorithm called GAL that adapts globally by sensing global congestion using injection queues at the source node. GAL routes the packets minimally at low load and on benign traffic and switches to non-minimal routing as the congestion is detected by the injection queues. But it has its own shortcomings. For example, its requirement for many injection queues makes it complex to implement. It also has very high latency when it starts routing traffic non-minimally [15]. Hence, an improved version of GAL, called ACQ, is proposed in [15]. Instead of using injection queues, ACQ estimates global congestion from the channel queues while relying on the implicit network back

pressure to transfer congestion information to these queues. ACQ uses a similar scheme with the Duato's algorithm to achieve deadlock freedom. This scheme devotes some amount of virtual channel resource for improbable deadlock situations. It is not a true fully-adaptive routing. And the global congestion information is inaccurate in ACQ.

In this paper, we introduce a new load-balanced routing algorithm called rHALB (r Hop Aware Load-Balanced) routing algorithm. rHALB uses time-out criteria to detect deadlocks, absorbs the detected deadlocked packet to the local node and retransmits it at a later time. It is also a true fully adaptive routing algorithm, which facilitates implementing load-balancing algorithms. rHALB uses the number of the detected potential deadlock packets to sense the congestion and then leads the packets to bypass the congested area as early as possible. rHALB is a limited-global-information-based routing algorithm, which is a compromise between global-information-based and local-information-based approaches. By using such information, rHALB requires a relatively simple process to collect and maintain link information in the neighborhood. Therefore, such an approach can be more cost-effective than those based on global or local information.

2 Notations

n_c the current node;

n^i_c neighbor node of n_c in the ith direction;

P packet set, p∈ P represents a single packet;

$S^{(p)}_{n_c}$ the set of directions in which p can follow minimal path form n_c to destination.

3 rHALB (r Hop Aware Load-Balanced) Routing Algorithm

3.1 Link State Vector and Direction Weight Vector

Before we introduce rHALB algorithm, we present some definitions here.

Definition 1. Let $G_{\text{k-ary n-cube}}=(\mathbb{N}_{\text{k-ary n-cube}}, \mathbb{C}_{\text{k-ary n-cube}})$, $n_c \in \mathbb{N}_{\text{k-ary n-cube}}$, Θ_{n_c} is link state vector for n_c and is defined as ($\theta^0_{n_c}, \theta^1_{n_c} \ldots \theta^i_{n_c} \ldots \theta^{d_n}_{n_c}$). $\theta^i_{n_c}$ is determined by the ToutPkCounter(n_c, i), which is associated with output physical channel i at node n_c. Each time a potential deadlocked packet is ejected from the network, the ToutPkCounter(n_c, i) is incremented for i∈ $S^{(p)}_{n_c}$. Once the packet is resent successfully, the ToutPkCounter(n_c, i) is decremented for i∈ $S^{(p)}_{n_c}$. Therefore, Θ_{n_c} can reflect the congestion degree of links around n_c.

Definition 2. Let $G_{\text{k-ary n-cube}}=(\mathbb{N}_{\text{k-ary n-cube}}, \mathbb{C}_{\text{k-ary n-cube}})$, $n_c \in \mathbb{N}_{\text{k-ary n-cube}}$ and p ∈ P, $\Phi_{n_c}(\mathrm{r},\mathrm{p})$ is direction weight vector (DWV) at n_c for packet p and can be defined as $(\varphi^0_{n_c}, \varphi^1_{n_c} \ldots \varphi^i_{n_c} \ldots \varphi^{d_n}_{n_c})$. $\Phi_{n_c}(\mathrm{r},\mathrm{p})$ is calculated by (1) as follows.

$$\varphi^i_{n_c}(\mathrm{r},\mathrm{p}) = \begin{cases} \theta^i_{n_c} & (\mathrm{r}=1) \\ \alpha\theta^i_{n_c} + (1-\alpha)[\beta \dfrac{\sum_{j\in S^{(\mathrm{p})}_{n_c}} \varphi^j_{n_c}(\mathrm{r}-1,\mathrm{p})}{|S^{(\mathrm{p})}_{n_c}|} + (1-\beta)\dfrac{\sum_{j\notin S^{(\mathrm{p})}_{n_c}} \varphi^j_{n_c}(\mathrm{r}-1,\mathrm{p})}{d_n - |S^{(\mathrm{p})}_{n_c}|}] & (1<\mathrm{r}\le \mathrm{k}/2) \\ \text{no sense} & (\mathrm{r}>\mathrm{k}/2) \end{cases} \quad (1)$$

where α, β are coefficients and $0 \le \alpha, \beta \le 1$. From (1), it is obvious that $\Phi_{n_c}(\mathrm{r},\mathrm{p})$ consists of two parts. The first part represents state of links connected to the current node. The second part shows the states of links around next node. If α is between 0.5 and 1, $\Phi_{n_c}(\mathrm{r},\mathrm{p})$ is mainly determined by the states of the connected links. Specially, for $\alpha = 1$, rHALB is completely based on local information. We use β to adjust the effect of non-minimal path on the direction weight vector (DWV). If β =1, only states of links in the minimal path are considered when calculating the second part of DWV.

3.2 The rHALB Algorithm

When the packet p arrives at the current node, rHALB calculates $\varphi^{\mathrm{i}}_{\mathrm{n_c}}(\mathrm{r},\mathrm{p})$ for each i. It always routes the packets along the shortest path with the increasing order of $\varphi^{\mathrm{i}}_{\mathrm{n_c}}(\mathrm{r},\mathrm{p})$. rHALB allows for non-minimal paths if all the channels on the shortest path are busy. A field in the packet is set to MisNum when the packet is generated. Each time the packet misroutes along the non-minimal path, MisNum is decremented by one. If MisNum is equal to zero, the packet could not take the non-minimal path anymore. A description of rHALB algorithm is shown as below.

rHALB algorithm:

1. Receive packet p;
2. Obtain the destination address and calculate $\Delta \mathrm{x_i}$;
3. For all i if $\Delta \mathrm{x_i} \neq 0$ Add i to Shortest-Dimension-Set;
4. If Shortest-Dimension-Set= $\varnothing$ Send the packet to the local node and Exit;
5. Calculate $\varphi^{\mathrm{i}}_{\mathrm{n_c}}(\mathrm{p})$;
6. Try to send p along i∈ Shortest-Dimension-Set with the increasing order of $\varphi^{\mathrm{i}}_{\mathrm{n_c}}(\mathrm{p})$;

```
7.  If succeed Exit;
8.  If for all i∈ Shortest-Dimension-Set the channels are busy
9.  {
10.  If θ^i_{n_c} <T_m and θ^i_{n_c^i} <T_m
11.     Wait until one of the channels in Shortest-Dimension-Set is idle.
12.  else
13.    {
14.     Obtain the MisNum from packet p;
15.     If (MisNum>0)
16.           {
17.                  Try to send the packet p along –i until Δx_i = 0
18.        If  succeed
19.           {
20.            MisNum--;
21.            Exit;
22.           }
23.     }
24.    else
25.     Wait until one of the channels in Shortest-Dimension-Set is idle.
26. }
27. }
28. Wait until timeout and deliver packet p to local node as a potential
    deadlocked packet;
29. Exit.
```

3.3 Deadlock Detection and Recovery

Deadlocks can be detected by a simple time-out criterion, which is similar to those suggested in [8, 10]. A TwaitCounter(n_c,i) is associated with physical channel i at node n_c. It is incremented every clock cycle. Hence it keeps tracks of the number of cycles during which the node n_c cannot send out the packet in the i^{th} direction. When TwaitCounter(n_c, i) is greater than the threshold T_{out}, the packet is ejected from the network as a potential deadlocked packet. When the packet is transmitted, either forwarded to the next node or ejected out of the network, TwaitCounter(n_c, i) is reset.

Instead of dropping the deadlocked packets [10], rHALB uses a software-based recovery mechanism, which is also used in [8,16]. By using such a mechanism, the potential deadlocked packet is absorbed by the local node and will be retransmitted at a later time.

Theorem 1. rHALB is livelock free and it can resolve every detected deadlock.

Proof. In most cases rHALB makes the packet choose the minimal path, thus providing livelock freedom. rHALB allows for the non-minimal path, but the number

of times for a packet to misroute is limited by MisNum. Therefore, after a finite number of hops, the packet will arrive at the destination. On the other hand, the rejected packet will be resent in finite time, which is proved in [9]. Thus, rHALB is livelock free.

Suppose that a deadlock has been detected. rHALB uses the software-based deadlock recovery mechanism to deliver the potential deadlocked packet to the local node, thus freeing the resource it occupied. Other packets that form the deadlock cycle can use the freed resources and continue the travel. Hence, the deadlock is resolved. □

4 Simulation Study

4.1 Evaluation Methodology

In this section, the performance of the rHALB algorithm will be evaluated by simulations and will be compared with those of dimension order (DO) routing algorithm, Duato's algorithm, ROMM, GAL and ACQ. The simulations have been done in the environment of OPNET simulation software [17]. Only virtual cut through switching mechanism is used in the simulations [18], but the algorithms are also suitable for wormhole switching [19] and store-and-forward switching.

The simulations are based on the following assumptions if not specially stated. Packet length distribution is a specific distribution SP (Size and Percent) that is based on the IP (Internet Protocol) packet size and percentages sampled over a two-week period [20]: 40 bytes (56 percent of all traffic), 1500 bytes (23 percent), 576 bytes (16.5 percent) and 52 bytes (4.5 percent). Such configurations of simulation environments are very close to the reality, which makes the results more convincing.

The traffic patterns used in the simulations include the uniform, the hotspot and the tornado traffic [2]. In the uniform traffic pattern, each node sends packets to all other nodes with the same probability. In the hotspot traffic pattern, one or more nodes are designated as the hotspot nodes, which receive hotspot traffic in addition to the regular uniform traffic. In the tornado traffic pattern, the node (i, j) only sends packets to node (i, $(j+\lceil k/2 \rceil -1) \bmod k$).

Low dimensional torus is popular in implementations and many routing algorithms have been proposed for these topologies [21]. Hence, a 2D torus network is used in the simulations. We have simulated an 8-ary 2-cube network and a 16-ary 2-cube network, however, only the results of the 8-ary 2-cube network are presented here due to space constraints. The results obtained for the 16-ary 2-cube network are similar to those of the 8-ary 2-cube network.

The nodes operate asynchronously and generate packets at time interval that follows the negative exponential distribution. For a fair comparison, three virtual channels are used for each algorithm. The value of **r** and MisNum for rHALB routing algorithm is 2 and 1 respectively.

Packets arriving at a destination node are consumed immediately. The performance of the routing algorithms is evaluated in terms of two main metrics: ETE (End To

End) delay and network throughput. The ETE delay is defined as the average time from the packet generation to the time when it reaches the destination. We use normalised throughput, which is equal to the number of packets that can be transmitted at the maximum load [22].

4.2 Simulation Results

As Fig.2 shows, under uniform traffic pattern for traffic loads less than 0.4 where little congestion is present, the six algorithms yield almost identical latency and throughput. When traffic loads increase to 0.5 and over, the performance gap starts to broaden. ROMM and ACQ are the first to saturate and yield the highest latency. The deterministic DO achieves similar network performance as those achieved by adaptive GAL and Duato's algorithm. The reason is that DO incorporates more long-term information about the characteristics of uniform traffic that may lead to more even distribution of traffic. But ROMM uses a random mediate node and ACQ uses non-minimal path, both of which break the evenness of the uniform traffic. rHALB turns out to be the best of the six algorithms, with the lowest latency and the highest throughput.

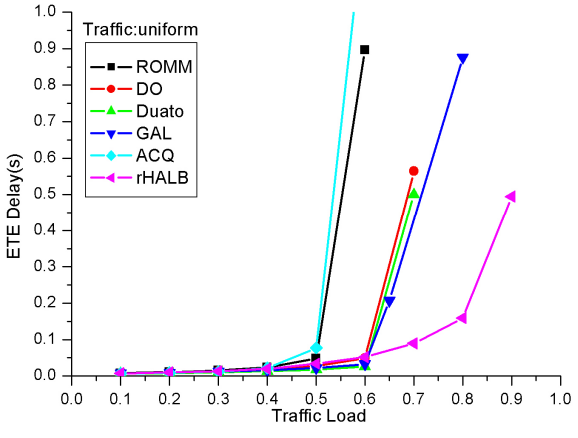


Fig. 2. Performance of the six algorithms under uniform traffic

We simulate hotspot traffic with two hotspots. The location of the hotspots can be randomly chosen because of the symmetry of the k-ary n-cube network. The hotspot receives 9% more traffic than the other nodes. As shown in Fig 3, the adaptive routing algorithms have advantages in balancing the network load. DO is the first to saturate and yields the highest ETE latency because it cannot distribute the traffic evenly among the links around the hotspots. It is obvious that rHALB outperforms the other routing algorithms in term of balancing the hotspot traffic. This is because rHALB can route the traffic to bypass the hotspot, thus lowering the burden of the hotspot node. Therefore, rHALB achieves a throughput three times higher than the deterministic algorithm with the same number of virtual channels. rHALB also achieves the lowest latency for the full range of traffic among the six algorithms.

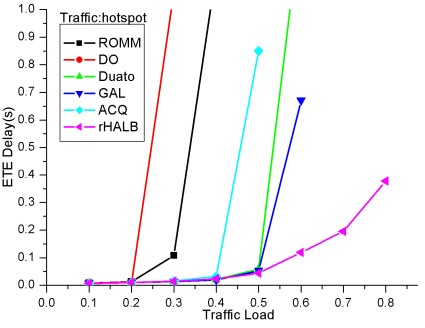


Fig. 3. Performance of the six algorithms under hotspot traffic

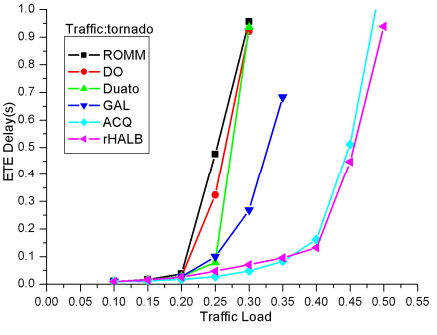


Fig. 4. Performance of the six algorithms under tornado traffic

The results obtained under the tornado traffic pattern are plotted in Fig.4. Tornado traffic pattern is one of the typical adversarial traffic patterns. In this pattern, the three minimal algorithms, DO, ROMM and Duato's algorithm show their shortcomings. They all saturate at around 0.25, which is 50% earlier than rHALB. The reason is that they route all of the packets in the direction of the minimal path, leaving the channels in the other direction idle. Compared with the minimal algorithms, GAL, ACQ and rHALB allow for the non-minimal path, thus utilize channels in the non-minimal paths. The load-balanced routing algorithms ACQ and rHALB offer the best performance because they can efficiently balance the load across the two directions in the y dimension.

5 Conclusion

A load-balanced routing algorithm called rHALB for k-ary n-cube networks is presented in this paper. rHALB makes use of limited global information, which is a tradeoff between global information and local information. When the traffic is low, rHALB routes all the packets minimally and thus obtains low latency and high throughput as seen in other minimal routing algorithms. When the traffic load is high

or when some adversarial traffic patterns are present, rHALB switches to non-minimal routing as congestion is detected by the direction weight vector. Hence, rHALB can make full use of links that left idle when minimal routing algorithms are used. Extensive simulations were carried out to compare rHALB with some other known algorithms such as DO, ROMM, GAL ACQ and Duato's algorithm in terms of throughput and latency under various network environments. The simulation results show that rHALB can delay saturation time and improve the overall performance. It provides the highest throughput and the lowest latency among the six algorithms on the three adversarial patterns.

Since we only simulated rHALB with r=2 and MisNum=1, in the future, we will study the effects of these two parameters on the performance of the rHALB algorithm. In addition, to apply rHALB algorithm to higher dimensional torus networks and other popular networks is another research topic in the future.

Acknowledgement

This research was supported by the Zhongxing Telecommunication Equipment Corporation (ZTE) Research Fund under Grant No. ZXJS200609120159.

References

1. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks, an Engineering Approach. Morgan-Kaufmann Press, San Francisco (2003)
2. Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan-Kaufmann Press, San Francisco (2004)
3. Bjerregaard, T., Mahadevan, S.: A Survey of Research and Practices of Network-on-Chip. ACM Computing Surveys 38(1) (2006)
4. Agarwal, A., Mustafa, M., Pandya, A.S.: Study of Network on Chip resources allocation for QoS. In: Canadian Conference on Electrical and Computer Engineering, pp. 1291–1295 (May 2006)
5. Pfister, G.: An introduction to the infiniband architecture. In: High Performance Mass Storage and Parallel I/O, IEEE Press, Los Alamitos (2001)
6. Sullivan, H., Bashkow, T.R.: A large scale, homogeneous, fully distributed parallel machine, I. In: Proc. of the International Symposium on Computer Architecture, pp. 105–117 (1977)
7. Duato, J.: A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. IEEE Trans. on Parallel and Distributed Systems 4, 1320–1331 (1993)
8. Anjan, K.V., Pinkston, T.M., Duato, J.: Generalized theory for deadlock-free adaptive routing and its application to Disha Concurrent. In: Proceedings of the 10th International Parallel Processing Symposium (April 1996)
9. Martínez, J.M., López, P.L., Duato, J., Pinkston, T.M.: Software-Based Deadlock Recovery Technique for True Fully Adaptive Routing in Wormhole Networks. In: ICPP 1997, pp. 182–189 (1997)
10. Kim, J., Liu, Z., Chien, A.: Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing. IEEE Trans. Parallel and Distributed Systems 8(3), 229–244 (1997)

11. Khonsari, A., Shahrabi, A., Ould-khaoua, M., Sarbazi-Azad, H.: Performance comparison of deadlock recovery and deadlock avoidance routing algorithms in wormhole-switched networks. IEE Proceedings-Computers and Digital Techniques, 150(2), 97–106 (2003)
12. Valiant, L.G.: A scheme for fast parallel communication. SIAM Journal on Computing 11(2), 350–361 (1982)
13. Nesson, T., Johnsson, S.L.: ROMM routing on mesh and torus networks. In: The Symposium on Parallel Algorithms and Architectures, Santa Barbara, CA, pp. 275–287 (1995)
14. Singh, A., Dally, W.J., Towles, B., Gupta, A.K.: Globally Adaptive Load-Balanced Routing on Tori. Computer Architecture Letters, 3 (March 2004)
15. Singh, A., Dally, W.J., Gupta, A.K., Towles, B.: Adaptive Channel Queue Routing on k-ary n-cubes. In: ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Barcelona, Spain (June 2004)
16. Suh, Y.J., Dao, B.V., Duato, J.: Software-Based Rerouting for Fault-Tolerant Pipelined Communication. IEEE Trans. Parallel and Distributed Systems 11(3) (March 2000)
17. Kermani, P., Kleinrock, L.: Virtual Cut through: A new computer communication switching technique. Computer Networks 3, 34 (1979)
18. Gu, H., et al.: Choice of Inner Switching Mechanisms in Terabit Router. In: Lorenz, P., Dini, P. (eds.) ICN 2005. LNCS, vol. 3420, pp. 826–833. Springer, Heidelberg (2005)
19. OPNET Modeler documentation, OPNET Technologies, Inc. (2007), http://www.opnet.com/
20. Newman, D.: Internet Core Router Test, On the Web (March 6, 2001) http://www.lightreading.com
21. Shih, J.-D.: Fault-tolerant wormhole routing in torus networks with overlapped block faults. IEE Proc. Comput. Digit. Tech. 150(1) (2003)
22. Towles, B., Dally, W.J.: Worst-case traffic for oblivious routing functions. In: SPAA 2002. 12th Annual ACM Symposium on Parallel Algorithms and Architectures, Canada (2002)

P2P File Sharing in Wireless Mesh Networks*

Luo Huiqiong[1], Ding Xuyang[1], Lao Hansheng[2], and Wang Wenmin[1]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 610054, China
[2] College of Zhongshan Torch Profession, Zhongshan, 528437, China

Abstract. In wireless mesh networks, the connective time would be relatively short in multi-hop nodes because of the mobility of peer nodes and the wireless collision. Therefore, it is important that how peers choose reasonable schemes to gain more benefits and improve the networks performance when they implement the peer-to-peer file sharing in limited environment. On the basis of the researches of peer-to-peer file sharing and the inherent characteristic of wireless mesh networks, a novel venture investment based file sharing model is proposed. The model provides a mechanism to maximize investment benefit and minimize investment venture, and makes optimization investment schemes through a benefit evaluation method. The simulation results show that the venture investment model can assist networks nodes to invest file sharing effectively and improve the performance of networks.

1 Introduction

The WMNs (Wireless Mesh Networks) is deployed with the end-to-end pattern, and it can be treated as a small wireless Internet according to its networks topology. In the Internet, the P2P (Peer-to-Peer) technique makes use of net-works resources, including computing resources, bandwidth resources, content resources and so on, to decrease the re-source requirements of the networks servers and bring a great deal of benefits in the same time. Recently, the computing and processing capability, storage capacity and the communication capability of mobile terminals are increased continuously. As the wide using of these equipments, it is possible to implement P2P File Sharing in WMNs.

In WMNs, the connective time of peers would be relative short because of the mobility of peer nodes and the wire-less collision. Therefore, it is important that how peers choose reasonable schemes to gain more benefits and to improve the networks performance when they implement the P2P file sharing in limited environment. The P2P file sharing model which is designed according to the characteristic of Internet can't apply to WMNs and how to implement the P2P file sharing in WMNs is still not be solved perfectly.

In this paper, we review some researches and applications related to the P2P file sharing technology at first. Then, according to the characteristics of the WMNs, we

* This research was supported by the National Natural Science Foundation of China (No. 60673142).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 402–413, 2007.

propose a venture investment based P2P file sharing model and simulate the model to prove its validity. Finally, we analyze the simulation results and draw a conclusion.

2 Related Works

You In the Internet, the primary protocols and models of P2P file sharing are Napster, BitTorrent, Gnutella, FastTrack, Chord, Freenet and so on. Napster [1] is a mixed and unstructured P2P file sharing meshwork. Central Server doesn't save files, but save the file index preserved by nodes, and clients search peers which preserved files for downloading. BitTorrent [2] is a mixed, unstructured, and multipoint-to-multipoint P2P file sharing networks, files or file blocks use the SHA1 Hash as marks, and the Hash can be used to verify the integrality of files. The central server is called as tracker, doesn't save sharing files, but save the information of the shared files and the shared user information. The information of shared files including tracker's address, file block size and file block hash which are saved in the files whose expanded name is torrent. These files usually promulgate via web. Gnutella [3] is a dispersed, unstructured P2P file sharing meshwork. Every Gnutella peer defines a local shared folder, these peers search files with part or the whole filename. The search is processed in flooding mode until to the scheduled layer. FastTrack [4] is an unstructured file sharing P2P networks with super nodes. When the peers start up, they register on the server to get the ID list of super nodes, then broadcast query the file store information in the whole networks via super nodes, and connect the peers to download files according to the usable file list returned form the query. Chord is a dispersed, structured P2P file searching protocol. In this protocol, every peer has its own virtual logical address. It composes a relative steady and close topological structure according to the addresses and constructs a DHT (Distributed Hash Table) to store files. Every search of peers searches the corresponding files according to the DHTs. Freenet [5] is a dispersed, loose-structured file sharing model. It doesn't have any central server, so the peers don't work with flooding mode as the Gnutella's. It only transmits request to the peers which seemly match with the request. If the matching is constituted, it makes sure the request chain and returns a response, then transmits files between two peers directly.

In WMNs, the P2P file sharing model designed according to the characteristic of the Internet is not suitable completely, because of the mobility of peer nodes and the wireless collision. Anna Hayes and others use the Gnutella protocol to implement the P2P file sharing [6], but the file query process of this model increases extra networks load, and it forms bottleneck when the higher degree nodes join in. All of these will affect the capability of the networks. Proem [7] defines four protocols to accomplish the reliable transfer, data sharing, synchronization, node verification, and peers search in the WMNs. But in fact this scheme is too broadly to actualize expected objectives. In many investigators' opinion, the primary barrier of implement the P2P file sharing is how to query files and lighten the load of queries, and they apply it to seek solving schemes [8,9,10,11,12]. But different with wire networks, only solving the problem of files and peers query can't make the P2P file sharing model applied in the WMNs normally. It is easy to induce networks congestion when transfers a mount of files

concurrently and continuously on multi-hop routes because of the inherent characteristic of WMNs, as it is shown in the simulations. Therefore, it is important that how peers choose reasonable schemes to gain more benefits and to improve the performance of networks while they implement the P2P file sharing in limited environment. If we regard the file sharing offered by one peer to other peers as the peer's contribution to the networks, then which files does the peer choose to store and how many benefits can be get from could be treated as a venture investment. And how to looking for the scheme of maximize investment benefit and minimize investment venture on the same risk level is the point. We built a file sharing model for the WMNs via the optimization venture investment in this paper.

3 Venture Investment Based File Sharing Model

Since the performance of the mobile nodes including computing, storage capacity, communication and so on increase continuously, many resources isn't exhausted by their owner in the WMNs. And the free resources can be used to help the server to store parts of files, and provide shared services to other peers. The essential spirit of P2P technology is cooperation and mutual benefit. The users provide their own resources in order to use others' resources and implements win-win finally. In general, the more pains, the more gains. But how many benefits will be get from the shared investment is uncertain. So, if the user can choose a wonderful scheme which the user can make more contributions to the networks, it can get the most expected income with the venture investment at the same risky level. And how to make a wonderful scheme is we need to solve.

3.1 Venture Investment Model

Suppose that every user have storage resources for venture investment. They store some files of the server to offer service to peers and expect to get the maximum benefit.

To build venture investment model, we have to quantify investment venture and income. According to the investment theory of Markowitz [13,14], the expected return can be weighed by the rate of return expectation, and the risk can be weighed by the square deviation. The higher the rate of return expectation is, the more the income is. While the more square deviation or standard deviation is, the more discrete degree of the rate of return is, in other words, the higher the uncertain of future income is. For describing the venture investment model and drawing investment scheme, we introduce symbols as follows:

$\tilde{r}_i$: It is the random income of unit storage resources while the file i is shared, $i = 1,2,\cdots,n$;

r_i : $E[\tilde{r}_i]$, it is the income expectation of unit storage resources while the file i is shared, $i = 1,2,\cdots,n$;

r_{n+1} : It is the income of unit storage resources while the user does not offer any file sharing service. Obviously, we can not expect get more returns than it sharing the files, so $r_i > r_{n+1}$, $i = 1,2,\cdots,n$;

σ_{ij} : It is the covariance $\mathrm{cov}(\tilde{r}_i, \tilde{r}_j)$ of $\tilde{r}_i$ and $\tilde{r}_j$, $i, j = 1,2,\cdots,n$;

p_i : It is the size of storage resources which is needed for sharing the file i, $i = 1,2,\cdots,n$, p_{n+1} is the size of the rest storage resources;

x_i : Whether the user shares the file i, 1 means yes while 0 means no, $i = 1,2,\cdots,n$, and the x_{n+1} denotes whether the user have rest storage resources, 1 means yes while 0 means no;

x_i^0 : It is the initial state of user coming into the networks, $i = 1,2,\cdots,n$, $x_{n+1}^0 \equiv 1$, and the meaning of value is same with x_i;

In order to evaluate the returns and risk of the scheme, we suppose as follows:

(1) There are $n(n = 1,2,\cdots)$ files stored on the file server. And the random incomes of unit storage resources of different files are not related.

(2) The expectation of the rate of returns is not known exactly, but lies in a known interval, $a_i \le r_i \le b_i, i = 1,2,\cdots,n$, where a_i and b_i are nonnegative constants. In fact, the a_i and b_i can be get from the real statistical results, and the choice of the venture investment has no relation to the values of a_i and b_i.

(3) The available storage resources hold the line during the investment process, marked as M^0.

With $y_i = \frac{p_i x_i}{M^0}$, $y_i^0 = \frac{p_i x_i^0}{M^0}$, we can get from the hypothesis (3):

$$\sum_{i=1}^{n+1} y_i = 1, \sum_{i=1}^{n+1} y_i^0 = 1 \tag{1}$$

From the formula (1), the random returns of investment $x = (x_1, x_2, \cdots, x_{n+1})$ are:

$$\sum_{i=1}^{n} \tilde{r}_i p_i x_i + r_{n+1} p_{n+1} x_{n+1} = M^0 [\sum_{i=1}^{n} \tilde{r}_i y_i + r_{n+1} y_{n+1}] \tag{2}$$

From the equation (2), the rate of random returns of investment $x = (x_1, x_2, \cdots, x_{n+1})$ is:

$$R(y) = \sum_{i=1}^{n} \tilde{r}_i y_i + r_{n+1} y_{n+1} \tag{3}$$

According to the formula (3), the expectation and square deviation of the rate of random returns are:

$$E[R(y)] = \sum_{i=1}^{n+1} r_i y_i \tag{4}$$

$$Var[R(y)] = \sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}y_i y_j \tag{5}$$

As a rational investor, his aim is the maximum returns expectation and the minimum risk square deviation. So he should make a trade-off between the two aims. We take ω and $1-\omega$ as the trade-off factors of $Var[R(y)]$ and $E[R(y)]$. The factor ω can be seen as the investor's risk aversion factor, and the bigger ω is, the more the investor detests risk. When ω=1, the investor is too conservative, and he only pays attention to the risk but forget the benefit; When ω=0, the investor is only looking for the benefit. As we know, a rational investor should detest the risk, so we suppose that: $0 < \omega \le 1$. Although the value of r_i isn't known exactly, we know a rational user usually reduces the risk on the basis of the maximum income. So, he has to solve the minimax problem shown in the formula (6).

$$P_\omega \begin{cases} \min\limits_{r}\max\limits_{y}[(1-\omega)\sum\limits_{i=1}^{n+1} r_i y_i - \omega\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\sigma_{ij}y_i y_j] \\ s.t.\begin{cases} \sum\limits_{i=1}^{n+1} y_i = 1 \\ a_i \le r_i \le b_i, i = 1,2,\cdots,n \\ 0 < \omega \le 1 \end{cases} \end{cases} \tag{6}$$

With $Y = \{y : \sum_{i=1}^{n+1} y_i = 1\}$, $R = \{r : a_i \le r_i \le b_i, i = 1,2,\cdots,n\}$. The above optimization problem can be written into a minimax problem (P_ω) in standard form as follow: $\min\limits_{r\in R}\max\limits_{y\in Y}[(1-\omega)\sum\limits_{i=1}^{n+1} r_i y_i - \omega\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\sigma_{ij}y_i y_j]$.

To solve the minimax problem (P_ω), we have to solve the problem $\max\limits_{y\in Y}[(1-\omega)\sum\limits_{i=1}^{n+1} r_i y_i - \omega\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\sigma_{ij}y_i y_j]$ at first, viz.:

$$\begin{cases} \max\limits_{y}[(1-\omega)\sum\limits_{i=1}^{n+1} r_i y_i - \omega\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\sigma_{ij}y_i y_j] \\ s.t.: \sum\limits_{i=1}^{n+1} y_i = 1 \end{cases} \tag{7}$$

With $y=(y_1,y_2,\cdots,y_n)^T$, $r=(r_1,r_2,\cdots,r_n)^T$, $I=(1,1,\cdots,1)^T \in R^n$, the optimization problem above can be changed into the unconstrained optimization problem in equation (8):

$$G=\max_{y}[(1-\omega)r_{n+1}+(1-\omega)(r-Ir_{n+1})^T y-\omega y^T V y] \tag{8}$$

According to the hypothesis (i), $\forall \sigma_{ij} \in V(\sigma_{ij})_{n\times n}$ if $i \neq j$,then $\sigma_{ij}=0$; $\forall \sigma_{ij} \in V(\sigma_{ij})_{n\times n}$, if $i=j$, then $\sigma_{ij}>0$. And for any non-vanishing real vector $X=(x_1,x_2,\cdots,x_n)^T$, $f(X)=X^T V X>0$ is true. So, the covariance matrix $V=(\sigma_{ij})_{n\times n}$ is a positive definite matrix. And because of $\omega>0$, we can get from the formula (8) :

$$\frac{\partial^2 G}{\partial y^2}=-2\omega V<0 \tag{9}$$

From the formula (9), the objective function of the problem G is a strictly concave function. Therefore, we can get the optimal solution when the first derivation of the object function equals to 0.

$$\frac{\partial G}{\partial y}=(1-\omega)(r-r_{n+1}I)-2\omega V y \tag{10}$$

The optimal solution is $y=\frac{1-\omega}{2\omega}V^{-1}(r-r_{n+1}I)$. So, the optimal value of the problem G's object function is:

$$(1-\omega)r_{n+1}+\frac{(1-\omega)^2}{4\omega}(r-r_{n+1}I)^T V^{-1}(r-r_{n+1}I) \tag{11}$$

With the original minimax problem and the expression (11), we can get the solution of the original problem (P_ω) via solving the problem shown in the formula (12):

$$\begin{cases}\min\limits_{r}(r-r_{n+1}I)^T V^{-1}(r-r_{n+1}I) \\ s.t.: a_i \le r_i \le b_i, i=1,2,\cdots,n\end{cases} \tag{12}$$

This minimization problem is a quadratic programming problem, and we can get the optimal solution via the simplex algorithm of linear programming when the value of $r_i(i=1,2,\cdots,n)$ is confirmed in the real environment. Suppose the optimal solution is: $r^*=(r_1^*,r_2^*,\cdots,r_n^*)^T$, the optimal solution of the original minimax problem (P_ω) is:

$$y_\omega = \frac{1-\omega}{2\omega} V^{-1}(r^* - r_{n+1} I) \tag{13}$$

From the equation (13), in the minimax sense, the expectation and the square deviation of the rate of returns are:

$$E_\omega[R(y)] = r_{n+1} + \frac{(1-\omega)}{2\omega}(r^* - r_{n+1} I)^T V^{-1}(r^* - r_{n+1} I) \tag{14}$$

$$Var_\omega[R(y)] = \frac{(1-\omega)^2}{4\omega^2}(r^* - r_{n+1} I)^T V^{-1}(r^* - r_{n+1} I) \tag{15}$$

Obviously, $E_\omega[R(y)]$ and $Var_\omega[R(y)]$ are both the decreasing function of ω, and the more the user detests the venture investment, the smaller the expectation and square deviation of the rate of returns are. We can see from the venture investment model that the user can find the optimal investment scheme to get the maximal income with the minimal venture if we can make sure the values of $\tilde{r}_i$ and r_i. Therefore, we need a relevant returns quantitative model of unit storage resource to confirm $\tilde{r}_i$ and r_i.

3.2 Returns Quantitative Model of Unit Storage Resource

Any investment benefit is the cash value of the future returns. There is a payoff ratio relation between the income and the former investment. So, we can constitute an income multiplier model according to the log of user's investment, and quantify the income of unit storage resource.

Income Multiplier (p) = History Income(B_H) / History Investment (I_H) (16)

The formula (16) is the conclusion of the history investment's payoff ratio, and it can be used as the quantify reference of current investment. If the user have not do any investment before, we set $p=1$. To evaluate r_i, we should evaluate two factors at first, the income multiplier p and the relative quotiety α of r_{n+1}. The quotiety should be set according to the condition of the networks by the user.

$$r_i = p \times (\alpha \times r_{n+1}) \tag{17}$$

The returns expectation calculated from the formula (17) can't reflect the future investment income of the WMNs accurately. There will be deviation more or less between the real return with the return expectation. The deviation of the investment returns mostly comes from that how many times does the shared files be used by other users, so it is reasonable that uses n as the expectation times to evaluate the income deviation. The value of n can be obtained according to both the complexion of other users' download requests and this user's subjective expectation. With β as the income deviation quotiety aroused by once request, the value of $\tilde{r}_i$ can be calculated from the formula (18).

$$\tilde{r}_i = (1+\beta \times n) \times r_i \tag{18}$$

A rational user won't choose the files which isn't used by others, so $n > 0$.

4 Simulation

To prove the validity of the venture investment model, we set an experiment environment which is similar to the real WMNs, and implement a contrastive simulation experiment between the central server download model and the venture investment based file sharing model.

4.1 Simulation Scenario

We choose the OPNET as simulation tool. The range is a rectangular domain which is 1000m×1000m, distributing 50 mobile nodes randomly. These nodes are classified into 4 kinds according to the investment storage resources, as shown in table1. The networks has one access point and the coordinates is (500, 500). It connects the server which offer FTP download. There are 60 files which can be downloaded on the server and are marked as F1 to F60, the size of every file is 50MB. The coverage radius of the wireless nodes communication is 250m, the data transfer velocity of every node is 2Mbps. During the simulation, we adopt the IEEE802.11 DCF model offered by OPNET as the MAC level of mobile nodes, and the DSR protocol as the router protocol.

Table 1. Nodes Distribution

Kinds of nodes	Available investment storage space	Node amount	Node marks
Large storage node (L)	150MB	5	H1 − H5
Middle storage node (M)	100MB	20	M1 − M20
Small storage node (S)	50MB	20	S1 − S20
None storage node (U)	0MB	5	U1 − U5

We set every node is none-selfish in simulation. However, there always are some selfish rational users in the real networks. They always attempt to expands their own return, and try to use more others users' resources, but offer less their own resources. In the application of P2P, it offers some punitive motivation mechanism to make users want to contribute more to the networks, and get more return [16,17]. We don't discuss the motivation mechanism in this paper.

To make the simulation more be similar to the real, we set the simulation parameters according to real networks, as shown in the table2.

Table 2. Simulation Factors

Mobile nodes	*Download request initiation time(sec)*	*Uniform Distribution (0, 600)*
	Download request time-out(sec)	10
	Time-out recall time	3
	Download request maximum restrict time	Unlimited
	Node velocity(m/s)	0~10(Random)
File server (FTP)	Data processing velocity (bytes/sec)	1,000,000
	Request response efficiency(request/sec)	1,000
	Service method	Client Requested
File slicing	File download from central server model	None
	Venture investment based file sharing model	25MB/Piece
Related quotiety α	$r_i : r_{n+1}$	1.5
Income deviation quotiety β	Per request	0.2

4.2 Simulation Results

The fig.1 and fig. 2 are the statistical results of traffic sent and packets dropped. We can see that characteristic of the venture based file sharing model is better than it of the central server download model. When the central server download model simulates to 150s, it causes congestion because of the operation increasing and networks resources lack. The networks congestion becomes worse in later 100s, the amount of traffic sent and packets dropped are increased suddenly, and the data packets are retransmitted constantly. The curve in fig.1 and the curve in fig.2 reflect this complexion very well. After 250s, the networks congestion is very serious in the central server download model, and the networks almost in paralysis state. Differently, the networks load is normal in the venture investment model. From the beginning of the simulation to 480s, the networks is always in normal state, and there is almost no packet lose. After 480s, the amount of traffic sent and packets dropped are in a rise trend, and the congestion comes out incidentally. But it's better than it of the central server download model, and the networks still can work normally.

Fig.3 shows the comparative results of the route hops of two models. The route hops are same at the beginning of the simulation, because there is no available shared files stored in the venture investment based file sharing model. So, the node should request the central FTP server to download. When simulation went 150s, the networks congestion appeared in the central server download model, and the congestion becomes more serious in later. Because some connections between nodes and the file

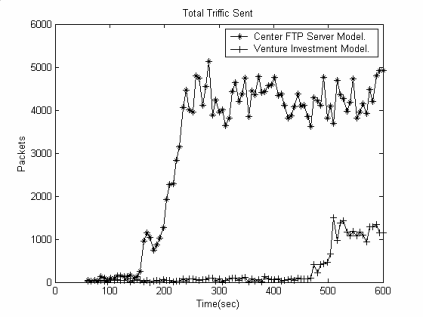


Fig. 1. Total traffic sent

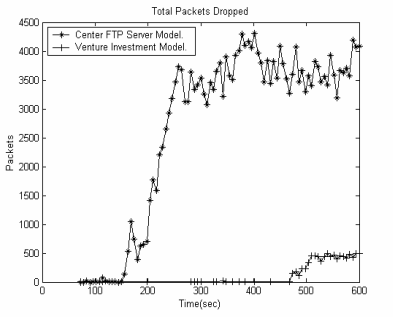


Fig. 2. Total packets dropped

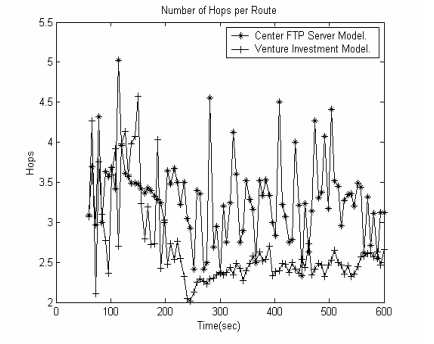


Fig. 3. Number of hops of route

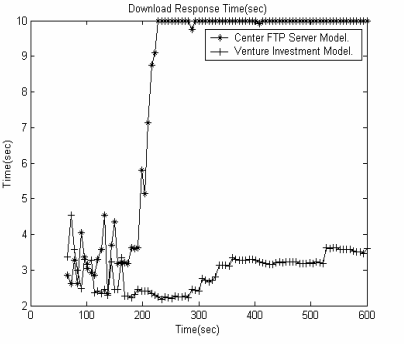


Fig. 4. Download response time

server break, these nodes search new routes to connect with the server continuously. It is why the route hops in the last stages is same with it in the initial stages of the simulation. While the nodes in the venture investment based file sharing model stored some files or file pieces, they can download the files from the neighborhood nodes which have stored them and needn't connect the central FTP server. Therefore, the route hops in the venture investment based file sharing model are less than it in the central server download model. The less route hops take advantage to the improvement of the networks performance.

The download request response time is shown in fig.4. From the beginning to 170s, the download response time in venture investment based file sharing model is similar with it of the central server download model. During this period, the primary download method is identical in these two models: the nodes almost download the files from the central FTP server. Later, the congestion appears in the central server download model. There are many data packets lost and retransmitted. It affects the networks performance badly. With the congestion, the request is flooded in many data packets and can't be responded in time. But in the venture investment based file sharing model, the request time is steady. Even if the light congestion appeared in 480s, the request response time is increased lightly, and it is in a normal acceptable range.

Above all we can conclude that the venture investment based file sharing model can decrease the load of central file server and the route hops between peers effectively. It can improve the networks performance.

5 Conclusion

In wireless mesh networks, the connective time would be relatively short in multi-hop peers because of the mobility of peer nodes and the wireless collision. Therefore, it is important that how peers choose reasonable schemes to gain more benefits and to improve the performance of networks while they implement the P2P file sharing in limited environment. On the basis of the researches of P2P file sharing and the inherent characters of wireless mesh networks, a novel venture investment based file sharing model is proposed. The model selects the optimize investment scheme for the nodes via the maximization income and the minimization venture of investment. The simulation results prove the validity of the model.

References

1. Byer, B., Martin, E., Edwards, C., et al.: Napster messages [EB/OL]. http://opennap.sourceforge.net/napster.txt
2. Cohen, B.: Bittorrent protocol specification [EB/OL]. http://bitconjurer.org/BitTorrent/protocol.html
3. Frankel, J.: The gnutella protocol specification v0.4 [EB/OL]. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
4. Liang, J., Kumar, R., Ross, K.W.: Understanding kazaa [EB/OL]. http://en.wikipedia.org/wiki/FastTrack
5. Clarke, I., Sandberg, O., Wiley, B., et al.: FreeNet: A distributed anonymous information storage and retrieval system [EB/OL]. http://www.ecse.rpi.edu/Hpmepages/shivkuma/teaching/sp2001/readings/freenet.pdf
6. Hayes, A., Wilson, D.: Peer-to-Peer Information Sharing in a Mobile Ad Hoc Environment. In: Proceedings of the Sixth IEEE Work-shop on Mobile Computing Systems and Applications (2004)
7. Schneider, J., Kortuem, G.: An application platform for mobile ad-hoc networks. In: Proceedings of the Workshop on Application Models and Programming Tools for Ubiquitous Computing (2001)
8. Schollmeier, R., Gruber, I., Finkenzeller, M.: Routing in mobile ad hoc and peer-to-peer networks, a comparison. In: International Workshop on Peer-to-Peer Computing, pp. 1–15 (2002)
9. Pucha, H., Hu, Y.C., Das, S.M.: Exploiting the synergy between peer-to-peer and mobile ad hoc networks. In: Proceedings of HotOS-IX: Ninth Workshop on Hot Topics in Operating Systems (2003)
10. Niethammer, F., Schollmeier, R., Gruber, I.: Protocol for peer-to-peer networking in mobile environments. In: ICCCN. Proceedings of IEEE 12th International Conference on Computer Communications and Networks (2003)
11. Hu, Y.C., Pucha, H., Das, S.M.: How to implement DHTs in mobile ad hoc networks? In: Poster in 10th ACM MOBICOM (2004)

12. Tang, B., Zhou, Z., Kashyap, A., Chiueh, T.-c.: Wireless And Mobile Computing, Networking And Communications. In: WiMob 2005. IEEE International Conference on, August 22-24, 2005, vol. 3, pp. 268–274 (2005)
13. Markowitz, H.: Portfolio selection. The Journal of Finance 7(1), 77–91 (1952)
14. Markowitz, H.: The optimization of a quadratic function subject to linear constraints. Naval Research Logistics Quarterly 3, 111–133 (1956)
15. Johnson, D.B., Maltz, D.A., Hu, Y.-C.: IETF MANET Working Group INTERNET-DRAFT, The Dynamic Source Routing Pro-tocol for Mobile Ad hoc Networks (DSR). Draft-ietf-manet-dsr-10 (July 19, 2004)
16. Feldman, M., Lai, K., Stoica, I., et al.: Robust incentive techniques for peer-to-peer networks [A]. In: Proceedings of the 5th ACM conference on Electronic commerce[C], New York, pp. 102–111 (2004)
17. Shneidman, J., Parkes, D.: Rationality and self-interest in peer-to-peer networks. In: International Workshop on Peer-to-Peer Systems (IPTPS) (2003)

General Biswapped Networks and Their Topological Properties*

Mingxin He[1,2], Wenjun Xiao[1], Weidong Chen[1], Wenhong Wei[1], and Zhen Zhang[1,2]

[1] Dept. of Computer Science, South China University of Technology Guangzhou, China 510641
[2] Dept. of Computer Science, Jinan University, Guangzhou, China 510632
mx.he@yeah.net, wjxiao@scut.edu.cn, chen_wei_dong@21cn.com, hquwwh@tom.com, zhang2003174@yahoo.com.cn

Abstract. In the paper, we propose a new systematic model, called General Biswapped Networks (GBSNs), to construct hierarchical interconnection networks that are able to maintain many desirable attributes of the underlying basis networks consisting of clusters. The model is inspired by and extended from Biswapped Networks (BSNs) and OTIS networks. A network in the new proposed model $Gbsw(\Omega, \Delta)$ is composed of m copies of some n-node basis network Ω and n copies of some m-node basis network Δ. Such a network uses a simple rule for connectivity that ensures its semi-regularity, modularity, fault tolerance, and algorithmic efficiency. In particular, the BSNs are special cases in which the two basis networks are the same. The Exchanged Hypercube is another example of GBSNs, i.e., $EH(s,t) \cong Gbsw(H(s), H(t))$. The proposed network model is able to reveal the intrinsic relation between Swapped Networks and OTIS architectures. We are to prove the homomorphic relation between GBSNs and the Cartesian product of its two basis networks. We also show the key topological parameters of GBSNs that are related to the parameters of its basis networks. We have obtained the results on inter-node distances, a general simple routing algorithm, Halmitonian cycles for networks composed of Halmitonian basis networks with even number of nodes or same number of nodes. Finally, this paper provides a new layout style for hierarchical interconnection networks that offers the researcher an opportunity to explore the topological properties of networks more intuitively and insightfully.

1 Introduction

An undirected graph is often used to model a processor/communication network in which the vertices (nodes) correspond to processor/communication units/ports and the edges (links/arcs) correspond to communication channels. Much work on interconnection networks can be categorized as ad hoc design and evaluation. Typically, a new interconnection scheme is often suggested and shown to be superior to some

* This research is supported by the Natural Science Foundation of Guangdong Province, China (No. 04020130).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 414–422, 2007.

previously studied network(s) with respect to one or more performance or complexity attributes. However, it will be more powerful if the network/graph researchers and engineers can use one or more type of simple graphs to construct a complicated network by a systematical approach. Different type of product graphs (such as Cartesian products, direct products, strong products, lexicographic products)[5], Hierarchical and Multistage Cayley graphs and coset graphs[1,2,4,9,11,12], especially semi-products and wreath products, are among the prime examples of systematic approach. The Optical Transpose Interconnection System (OTIS)[3,7], Index-Permutation Graph Model[14], Swapped Networks[8,13], Biswapped Networks (BSNs)[10] are the examples of some specific models that are used to systematically compose hierarchical interconnection networks.

In this paper, we propose a new systematic model, called General Biswapped Networks (GBSNs), to compose hierarchical interconnection networks. The model is inspired by and extended from Biswapped Networks that are related to swapped networks or OTIS networks, which have been studied by a number of researchers [3,7,8,13]. A network in the newly proposed model $Gbsw(\Omega, \Delta)$ is composed of m copies of some n-node basis network Ω and n copies of some m-node basis network Δ. The process uses a simple rule for connectivity, called biswapping strategy, that ensures its semi-regularity, modularity, fault tolerance, and algorithmic efficiency. In particular, the BSNs are the special cases in which the two basis networks are the same. The Exchanged Hypercube investigated in [6] is another specific example of GBSNs. Thus, the work presented here extends and generalizes the systematic method BSNs proposed in [10] for the construction of large, scalable, modular, and robust parallel architectures, while maintaining many desirable attributes of the underlying basis networks that comprises its clusters.

The rest of this paper is organized as follows: First, in Section 2, we describe the model of GBSNs and discuss its relations to known network models by examples in section 2; Then, in Section 3, we show how the key topological parameters of a GBSN are related to the parameters of its basis networks and a simple routing algorithm is developed; In Section 4, we show Hamiltonicity of GBSNs and in Section 5 we conclude this paper with some discussions.

2 Definitions and Relations to Known Network Models

Let Ω be any undirected graph (thereafter, use graph to indicate undirected graph for conciseness) with the vertex set $V(\Omega) = \{g_1, g_2, \ldots, g_n\}$ and the arc set $E(\Omega)$. And let Δ be any graph with the vertex set $V(\Delta) = \{h_1, h_2, \ldots, h_m\}$ and the arc set $E(\Delta)$. The General Biswapped Network $Gbsw(\Omega, \Delta) = \Sigma = (V(\Sigma), E(\Sigma))$ is a graph composed of m copies of Ω, $\{\Omega_1, \Omega_2, \ldots, \Omega_m\}$, and n copies of Δ, $\{\Delta_1, \Delta_2, \ldots, \Delta_n\}$, with additional external links in between. The vertex and edge sets of Σ is specified as:

$$V(\Sigma) = \{g_{ij} \mid g_{ij} \in \Omega_i, i=1,2,...,m, j=1,2,...,n\} \cup \{h_{ji} \mid h_{ji} \in \Delta_j, i=1,2,...,m, j=1,2,...,n\}$$

$$E(\Sigma) = \{ (g_{iu}, g_{iv}) \mid (g_u, g_v) \in E(\Omega), i=1,2,...,m, u=1,2,...,n, v=1,2,...,n \} \cup$$
$$\{ (h_{js}, h_{jt}) \mid (h_s, h_t) \in E(\Delta), j=1,2,...,n, s=1,2,...,m, t=1,2,...,m \} \cup$$
$$\{ (g_{ij}, h_{ji}) \mid g_{ij} \in \Omega_i, h_{ji} \in \Delta_j, i=1,2,...,m, j=1,2,...,n \}$$

Intuitively, the definition postulates two parts: m clusters of Ω (part 0) and n clusters of Δ (part 1) with inter-cluster links. Denote g_{ij} as <0, i, j> and h_{ji} as <1, j, i>, representing <*part*#, *cluster*#, *node*#>, for convenience. The edge set of Σ is composed of three parts: intra-cluster edges in Ω_i (part 0), intra-cluster edges in Δ_j (part 1), and inter-cluster or swapping edges between two parts.

The name "General Biswapped Network" (GBSN) arises from the definition just introduced: when clusters Ω_i and Δ_j are viewed as supernodes, the resulting graph of supernodes is a complete m+n-node bipartite graph $K_{m,n}$, and the inter-cluster links connect nodes between two parts in which the cluster number and the node number within cluster are interchanged or swapped.

From the above definition, it is clearly that Ω and Δ plays a symmetric role in the composition of $Gbsw(\Omega, \Delta)$. Denote $A \cong B$ as graph A is isomorphic to graph B, we have

Proposition 1. $Gbsw(\Omega, \Delta) \cong Gbsw(\Delta,\Omega)$.

Proof: It stands for the fact that the map from the node <i, c, v> in $Gbsw(\Omega, \Delta)$ to the node <1-i, c, v> in $Gbsw(\Delta,\Omega)$, in which $i \in \{0,1\}$, is a isomorphism. □

Fig. 1 shows some specific examples of GBSNs, in which (a) is $Gbsw(K_2, C_3)$. The new layout styles shown in Fig. 1 for hierarchical interconnection networks is originally created by the authors. We believe that a felicitous layout of networks can benefit investigators to explore the topological properties of networks more intuitively and insightfully.

In particular, the Biswapped Networks (BSNs), which is investigated in [10], is the special case of GBSNs in which the two basis networks are the same, i.e., $Bsw(\Omega) \cong Gbsw(\Omega,\Omega)$. It is known that if the basis network Ω is a Cayley graph, so is

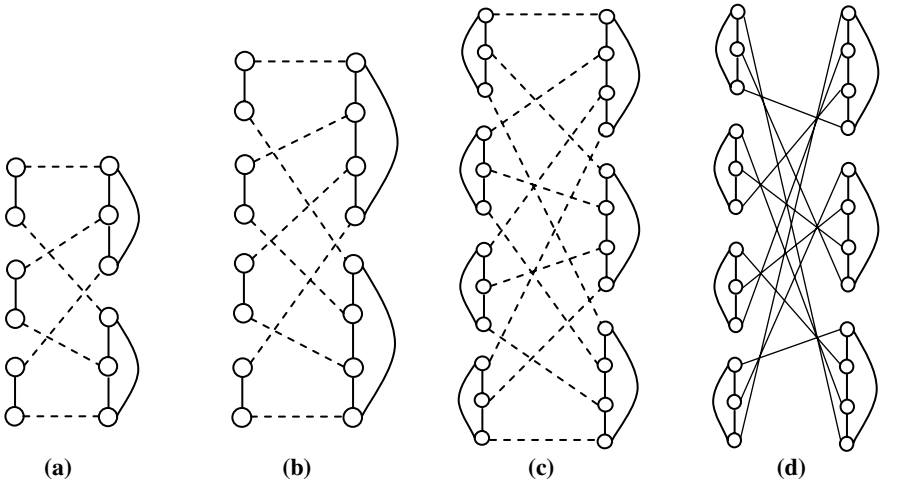


Fig. 1. Examples of GBSNs. **(a)** $Gbsw(K_2, C_3)$ **(b)** $EH(1,2) \cong Gbsw(H(1), H(2)) \cong Gbsw(K_2, C_4)$ **(c)** $Gbsw(C_3, C_4)$ **(d)** $G_{OTIS}(C_3,C_4) \cong Gbsw(C_3, C_4)$.

Bsw(Ω) and the swapped networks (also called OTIS) with self loop in every non-inter-cluster node are coset graphs of the related biswapped networks.

The Exchanged Hypercube investigated in [6] is another specific example of GBSNs. In fact an $EH(s, t)$ is composed of s copies of hypercube $H(t)$ and t copies of hypercube $H(s)$ with inter-cluster links, i.e., $EH(s, t) \cong Gbsw(H(s), H(t))$. Fig. 1(b) shows $EH(1, 2) \cong Gbsw(H(1), H(2)) \cong Gbsw(K_2, C_4)$.

If we use another different rule to establish the inter-cluster links, more interesting hierarchical interconnection networks can be constructed. For example, if we keep all the "supernodes" unchanged, and just change our biswapping strategy to an OTIS style for those inter-cluster links, i.e., change the third part of links in $E(\Sigma)$ to

$$\{ (g_{ij}, h_{m+1-j,\, n+1-i}) \mid g_{ij} \in \Omega_i,\ h_{m+1-j,\, n+1-i} \in \Delta_{m+1-j}\ ,\ i=1,2,...,m,\ j=1,2,...,n \}$$

Then we get a new hierarchical interconnection model, i.e., General OTIS Networks, and denote it to $G_{OTIS}(\Omega, \Delta)$.

Fig. 1(c) shows $Gbsw(C_3, C_4)$ and (d) shows $G_{OTIS}(C_3, C_4)$. It is evidently that $Gbsw(C_3, C_4) \cong G_{OTIS}(C_3, C_4)$.

In fact, above two interconnection models construct the same interconnection networks for either Ω or Δ shows a little bit symmetric properties.

Proposition 2. *For any graph Ω and Δ, if either there is an auto-isomorphic from* $\{ g_1, g_2, \ldots, g_n \}$ *to* $\{ g_n, g_{n-1}, \ldots, g_1 \}$ *for Ω, or there is an auto-isomorphic from* $\{ h_1, h_2, \ldots, h_m \}$ *to* $\{ h_m, h_{m-1}, \ldots, h_1 \}$, *then* $Gbsw(\Omega, \Delta) \cong G_{OTIS}(\Omega, \Delta)$.

Proof: It is evidently from comparing the definition of $Gbsw(\Omega, \Delta)$ with that of $G_{OTIS}(\Omega, \Delta)$. □

In another way, if we rename the node names from $\{g_1, g_2, \ldots, g_n\}$ to $\{g_n, g_{n-1}, \ldots, g_1\}$ for any n-node graph Ω and get Ω', then $Gbsw(\Omega, \Delta) \cong G_{OTIS}(\Omega', \Delta)$. Similarly, we have $Gbsw(\Omega, \Delta) \cong G_{OTIS}(\Omega, \Delta')$.

Remark 1. Proposition 2 reveals the intrinsic relation between Swapped Networks and OTIS architectures, which were investigated by researchers came from different domains. This is the reason why Pharhami said in [8] that "It was recently pointed out to the author by an anonymous reviewer that swapped networks are the same as optical transpose interconnection system (OTIS) architectures which have been extensively studied by other researchers. "

In the last of this section, we show the homomorphic relation between GBSNs and the Cartesian product of its two basis networks, so that GBSNs may share some topological properties of the related Cartesian products[5].

Proposition 3. *Gbsw*(Ω, Δ) *is homomorphic to the Cartesian product of* Ω *and* Δ, *i.e.*,Ω□Δ, *with a self-loop on each node*.

Proof: The Cartesian product $\Omega \square \Delta = \Pi = (V(\Pi), E(\Pi))$ of two graphs Ω and Δ is defined as follows:

$$V(\Pi) = V(\Omega) \times V(\Delta) = \{ \langle g_i, h_j \rangle \mid i=1,2,...,m,\ j=1,2,...,n \}$$
$$E(\Pi) = \{ (\langle g_u, h_s \rangle, \langle g_v, h_t \rangle) \mid g_u = g_v\ , (h_s, h_t) \in E(\Delta), \text{ or, } (g_u, g_v) \in E(\Omega), h_s = h_t \}$$

Recall $V(Gbsw(\Omega, \Delta) = \Sigma) = \{ g_{ij}, h_{ji} \mid g_{ij} \in \Omega_i\ , h_{ji} \in \Delta_j\ , i=1,2,...,m,\ j=1,2,...,n \}$, and consider the map f from $V(\Sigma)$ to $V(\Pi)$: $g_{ij} \rightarrow \langle g_i, h_j \rangle$, $h_{ji} \rightarrow \langle g_i, h_j \rangle$. It is clearly that f

keeps all the intra-cluster connectivity from $E(\Sigma)$ to $E(\Pi)$, and exactly maps every inter-cluster edge (g_{ij}, h_{ji}) in $E(\Sigma)$ to the self-loop on node $<g_i, h_j>$. This implies f is a homomorphism from Σ to Π with self-loops. □

3 Topological Properties and Routing Algorithm

Let's fix some notations in what follows. For any graph Γ, the number of its nodes is denoted as $|\Gamma|$. The degree of a node g in Γ is $deg_\Gamma(g)$. The distance between nodes g_1 and g_2 in Γ is given by $dist_\Gamma(g_1, g_2)$. The diameter of Γ, i.e., the maximum distance between any two nodes in Γ, is $D(\Gamma)$.

We first show the basic topological parameters of GBSNs related to the parameters of its basis networks.

Theorem 1. *Let* $\Sigma = Gbsw(\Omega, \Delta)$, $|\Omega| = n$, $|\Delta| = m$, i=1,2,...,m, j= 1,2,...,n. *Then:*

(1) $|\Sigma| = 2\,|\Omega|\,|\Delta| = 2mn$

(2) $deg_\Sigma(g_{ij}) = deg_\Omega(g_j) + 1$ *and* $deg_\Sigma(h_{ji}) = deg_\Delta(h_i) + 1$

(3) $dist_\Sigma(g_{iu}, g_{iv}) \le dist_\Omega(g_u, g_v)$, u=1,2,...,n, v=1,2,...,n *and* $dist_\Sigma(h_{js}, h_{jt}) \le dist_\Delta(h_s, h_t)$, s=1,2,...,m, t=1,2,...,m

(4) $dist_\Sigma(g_{su}, g_{tv}) \le dist_\Omega(g_u, g_v) + dist_\Delta(h_s, h_t) + 2$, $s \ne t$ *and* $dist_\Sigma(h_{us}, h_{vt}) \le dist_\Omega(g_u, g_v) + dist_\Delta(h_s, h_t) + 2$, $u \ne v$

(5) $dist_\Sigma(g_{su}, h_{vt}) \le dist_\Omega(g_u, g_v) + dist_\Delta(h_s, h_t) + 1$

(6) $D(\Sigma) \le D(\Omega) + D(\Delta) + 2$.

Proof: (1) and (2) are evidently by the definition of $Gbsw(\Omega, \Delta)$ given in section 2.

(3) stands for excepting the intra-cluster shortest path in Ω or Δ, there may have another path connecting two intra-cluster nodes through inter-cluster link that is shorter than the intra-cluster shortest path.

(4) is true for that for any two nodes g_{su}, g_{tv} on the different clusters in the same part, there are two paths connecting them in Σ: $g_{su} \to g_{sv} \to h_{vs} \to h_{vt} \to g_{tv}$ and $g_{su} \to h_{us} \to h_{ut} \to g_{tu} \to g_{tv}$. Both paths have the same length $dist_\Omega(g_u, g_v) + dist_\Delta(h_s, h_t) + 2$. Similarly, the $dist_\Sigma(h_{us}, h_{vt})$ inequality stands.

(5) stands for the fact that there is a path between g_{su} and h_{vt} in Σ: $g_{su} \to g_{sv} \to h_{vs} \to h_{vt}$ with a length of $dist_\Omega(g_u, g_v) + dist_\Delta(h_s, h_t) + 1$.

(6) is a corollary from (3), (4) and (5). □

Remark 2. As a conjecture, the equality in (3), (4), (5) and (6) in Theorem 1 stands for GBSNs with regular, connected and symmetric basis graphs. It has been proved for the biswapped networks.

Based on the proof of Theorem 1, we can easily obtain a general and simple routing algorithm for a GBSN, assuming the availability of the routing algorithms for the basis networks Ω and Δ. Assume that $next_\Omega(g_1, g_2)$ and $next_\Delta(h_1, h_2)$ are local functions representing the corresponding routing algorithms to obtain the first intermediate node in the routing path from g_1 to g_2 in Ω or from h_1 to h_2 in Δ respectively. Recall the unified notation $<i, c, v> = Node$ for all nodes in Σ. Then, the algorithm shown below in a C-like language can be used to derive the first intermediate node on an almost-shortest routing path in Σ from the source $<i, c_1, v_1>$ to the target $<j, c_2, v_2>$.

Algorithm 1. Routing algorithm for $\Sigma = Gbsw(\Omega, \Delta)$:

```
Node next_Σ ( <i, c1, v1>, <j, c2, v2> ) {
  if i == j {      // i = j, routing in the same part
    if c1 == c2 && v1 == v2     // destination has been reached
      return <i, c1, v1>
    else if v1 == v2      // c1 ≠ c2, change to the opposite part
      return <1-i, v1, c1>
    else                  // keep in the same cluster
      return <i, c1, (i==0) ? next_Ω(v1, v2) : next_Δ(v1, v2)>
  } else {      // i ≠ j, routing between parts
    if v1 == c2           // change to the opposite part
      return <1-i, v1, c1>
    else                  // keep in the same cluster
      return <i, c1, (i==0) ? next_Ω(v1, c2) : next_Δ(v1, c2)>
  }
}
```

Remark 3. For most cases, in specially those with regular and symmetric basis graphs, the routing algorithm for GBSNs given above is optimal. The modifier "almost" put before "shortest" implies that there has possibility in cases that the exact inequality in (3), (4) or (5) of Theorem 1, i.e., $<$, stands.

4 Halmiltonicity of GBSNs Built of Halmiltonian Basis Networks

A Hamiltonian cycle in a graph is a cycle that visits each node exactly once. A graph is Hamiltonian if it contains a Hamiltonian cycle. Hamiltonicity is a useful property for interconnection networks. One of our main results in this paper show that if the basis graphs Ω and Δ are Hamiltonian, either both Ω and Δ have even number of nodes, or, have the same number of nodes, then the resulting GBSN is Hamiltonian.

Theorem 2. *If the basis graphs Ω and Δ both have even number of nodes and are Hamiltonian, then so is the graph $Gbsw(\Omega, \Delta) = \Sigma$.*

Proof: Assume $|\Omega| = n = 2s$, $|\Delta| = m = 2t$, the Halmilton cycles in Ω and Δ are $\{g_1g_2 \ldots g_ng_1\}$ and $\{h_1h_2 \ldots h_mh_1\}$ respectively. The Hamiltonicity of Σ will be proved by the following systematic constructing steps:

Step 1, Select all the inter-cluster edges and the following intra-cluster edges:

$\{ (\langle 0, i, 2x\text{-}1\rangle, \langle 0, i, 2x\rangle) \mid i=1,2,...,m,\ x=1,2,...,s \} \cup$

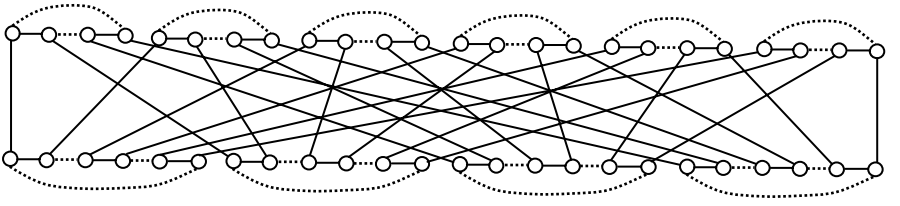
$\{ (\langle 1, j, 2y\text{-}1\rangle, \langle 1, j, 2y\rangle) \mid j=1,2,...,n,\ y=1,2,...,t \}$

The selected edges with all the nodes in Σ compose *st* of 8-node sub-cycles.

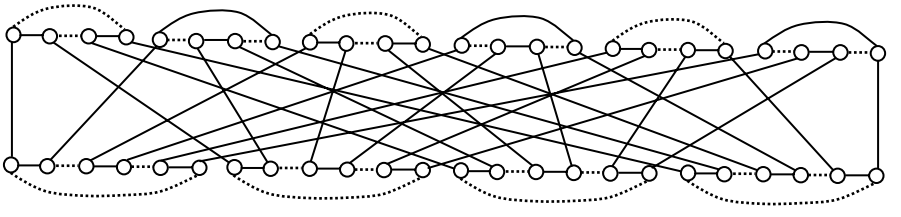
Step 2, Choose either part, e.g. part 0, break the *s* of 8-node sub-cycles on each cluster with even number by unselecting the intra-cluster edges, and merge them by reselecting the following edges:

$\{ (\langle 0, 2y, 2x\rangle, \langle 0, 2y, (2x+1) \bmod n\rangle) \mid x=1,2,...,s,\ y=1,2,...,t \}$

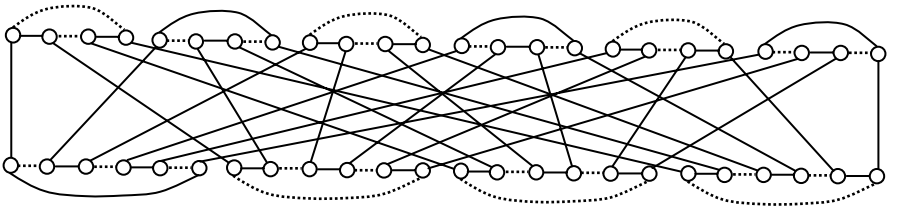
The selected edges with all the nodes in Σ compose t of sub-cycles with $8s$ nodes respectively.



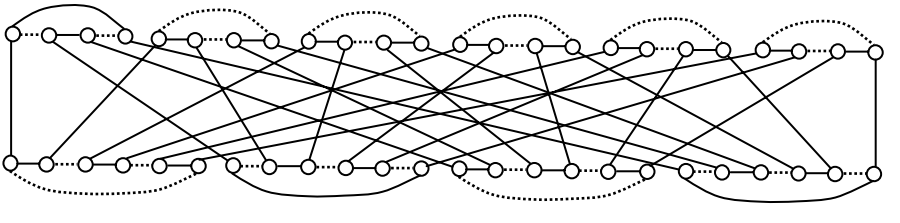
(a) Step 1: Find $st = 2 \times 3 = 6$ of 8-node sub-cycles in $Gbsw(C_4, C_6)$



(b) Step 2: Find $t = 3$ of sub-cycles with $8s = 16$ nodes respectively in $Gbsw(C_4, C_6)$



(c) Step 3: A Hilmilton cycle with $8st = 8 \times 2 \times 3 = 48$ nodes in $Gbsw(C_4, C_60029$



(d) Another Hilmilton cycle in $Gbsw(C_4, C_6)$

Fig. 2. Constructing Halmilton Cycles in $Gbsw(C_4, C_6)$

Step 3, Choose the opposite part used in step 2, i.e., part 1, break the t of 8s-nodes sub-cycles on any cluster, e.g., on cluster 1, by unselecting the intra-cluster edges, and merge them by reselecting the following edges:

$\{ (<1, 1, 2y>, <1, 1, (2y+1) \ mod \ m>) \mid y=1,2,...,t \}$

The selected edges with all the nodes in Σ compose a cycle with 8st nodes, which is a Halmilton cycle in Σ. □

Fig. 2(a) shows the result of $Gbsw(C_4, C_6)$ after step 1, in which there are 6 of 8-node sub-cycles; (b) shows the result of $Gbsw(C_4, C_6)$ after step 2, in which there are 3 of 16-node sub-cycles; (c) shows a Halmilton cycle in $Gbsw(C_4, C_6)$ after step 3.

Fig. 2 (d) shows another Halmilton cycle in $Gbsw(C_4, C_6)$ reached by merging sub-cycles on the opposite part.

Remark 4. The three constructing steps used in the proof of Theorem 2 can be easily implemented by an algorithm to find Halmilton cycles in those GBSN automatically.

Theorem 3. *If the basis graphs* Ω *and* Δ *both have the same number of nodes and are Hamiltonian, then* $Gbsw(\Omega, \Delta) = \Sigma$ *is Hamiltonian.*

Proof: The proof is similar as the proof of Theorem 3 in [10] on the Halmiltonicity of the swapped networks $Bsw(\Omega) \cong Gbsw(\Omega,\Omega)$. Omitted for conciseness. □

Remark 5. For a GBSN built of two unbalanced Hamiltonian basis graphs of at least one with odd number of nodes, whether it contains a Hamiltonian cycle is an open problem. In fact, there is no Hamiltonian cycle in $Gbsw(K_2,C_3)$ shown in Fig. 1(a).

5 Conclusions

In this paper, we have provided a number of important results on the new hierarchical interconnection model called "General Biswapped Networks (GBSNs)". A network constructed by the new model $Gbsw(\Omega, \Delta)$ is composed of m copies of some n-node basis network Ω and n copies of some m-node basis network Δ, using a simple biswapping strategy. The strategy ensures semi-regularity, modularity, fault tolerance, and algorithmic efficiency of the constructed network. The work extends and generalizes the systematic method called Biswapped Networks (BSNs) proposed in [10] to construct large, scalable, modular, and robust parallel architectures, while maintaining many desirable attributes of the underlying basis networks that comprises its clusters.

We have discussed the relations between GBSNs and a series of interconnection models including OTIS, Swapped Networks, Biswapped Networks, the Exchanged Hypercube and Cartesian products of its two basis graphs. This model is able to reveal the intrinsic relation between Swapped Networks and OTIS architectures. We also proved the homomorphic relation between GBSNs and the Cartesian product of its two basis networks. We showed key topological parameters of GBSNs that are related to the parameters of its basis networks. We also obtained results on inter-node distances, a simple almost-optimal routing algorithm, Halmitonian cycles for GBSNs built of Halmitonian basis networks with even number of nodes or same number of nodes. Finally, we provide a new layout style for hierarchical interconnection

networks that can help investigators to explore the topological properties of networks more intuitively and insightfully.

Because of the generality of this model and the related theorems, we expect that they will find many more applications beyond what have been discussed in this paper. We are currently investigating the applications of our method to the problems related to routing as well as the average inter-node distance in certain subclasses of our networks. These results, along with potential applications in the following areas will be reported in future:

- Load balancing and congestion control
- Scheduling and resource allocation
- Fault tolerance and graceful degradation

These research topics constitute important practical challenges in the design, evaluation, and efficient operation of parallel and distributed computer systems.

References

1. Akers, S.B., Krishnamurthy, B.: A Group Theoretic Model for Symmetric Interconnection Networks. IEEE Trans. Computers 38, 555–566 (1989)
2. Annexstein, F., Baumslag, M., Rosenberg, A.L.: Group Action Graphs and Parallel Architectures. SIAM J. Computing 19, 544–569 (1990)
3. Day, K., Al-Ayyoub, A.: Topological Properties of OTIS-Networks. IEEE Trans. Parallel and Distributed Systems 13(4), 359–366 (2002)
4. Heydemann, M.: Cayley Graphs and Interconnection Networks. In: Graph Symmetry: Algebraic Methods and Applications, pp. 167–224 (1997)
5. Imrich, W., Klavzar, S.: Product Graphs, Structure and Recognition. John Wiley & Sons, New York (2000)
6. Loh, P.K.K., Hsu, W.J., Pan, Y.: The Exchanged Hypercube. IEEE Trans. Parallel and Distributed Systems 16(9), 866–874 (2005)
7. Marsden, G., Marchand, P., Harvey, P., Esener, S.: Optical Transpose Interconnection System Architectures. Optics Letters 18(13), 1083–1085 (1998)
8. Parhami, B.: Swapped interconnection networks: Topological, performance, and robustness attributes. J. Parallel Distrib. Comput. 65, 1443–1452 (2005)
9. Sabidussi, G.: Vertex-transitive graphs. Monatsh. Math. 68, 426–438 (1964)
10. Xiao, W.J., Chen, W.D., He, M.X., Wei, W.H., Parhami, B.: Biswapped Networks and Their Topological Properties. In: SNPD 2007. Proc. of the 8th Int. Conf. on SE, AI, Networking, and Parallel/Distrib. Comput, Qingdao, China, vol. 2, pp. 193–198 (2007)
11. Xiao, W.J., Parhami, B.: Some Mathematical Properties of Cayley Digraphs with Applications to Interconnection Network Design. Int. J. Computer Mathematics 82, 521–528 (2005)
12. Xiao, W.J., Parhami, B.: Further Mathematical Properties of Cayley Digraphs Applied to Hexagonal and Honeycomb Meshes. Discrete Applied Mathematics (to appear)
13. Yeh, C.H., Parhami, B.: Swapped Networks: Unifying the Architectures and Algorithms of a Wide Class of Hierarchical Parallel Processors. In: Proc. Int. Conf. Parallel and Distributed Systems, pp. 230–237 (1996)
14. Yeh, C.H., Parhami, B.: The Index-Permutation Graph Model for Hierarchical Interconnection Networks. In: Proc. of the Int. Conf. on Parallel Processing, Aizu, Japan, pp. 48–55 (1999)

Design a Hierarchical Cache System for Effective Loss Recovery in Reliable Multicast

Zhijun Wang, Xiaopeng Fan, and Jiannong Cao

Department of Computing,
The Hong Kong Polytechnic University, Hong Kong
{cszjwang,csxpfan,csjcao}@comp.polyu.edu.hk

Abstract. Packet loss recovery is a key issue in reliable multicast. An effective way for packet loss recovery is to place repair servers with active routers along the transmission paths. These repair servers naturally form a hierarchical cache system due to the hierarchical nature of the multicast tree. How to design an effective hierarchical cache system to minimize the packet loss is important. In this paper, we first derive a cooperative caching efficiency model for a hierarchical cache system. Based on the model, a heuristic Cooperative Cache Replacement (CCR) algorithm is proposed to achieve efficient cache performance for reliable multicast systems. The implementation issues are also discussed in detail. The ns-2 based simulations are conducted to evaluate the performance of the proposed algorithm by compared to the optimal caching time (OCT) based algorithm. The results show that CCR effectively reduces the packet loss recovery latency.

Keywords: Reliable multicast, loss recovery, cooperative cache, replacement algorithm.

1 Introduction

Packet loss recovery is important in reliable multicast. One effective way for packet loss recovery is to place repair servers in active routers along the transmission paths [7] [8]. Each repair server allocates certain of buffer space to cache received packets for possible loss recovery. These repair servers intrinsically form a hierarchical cache system due to the hierarchical nature of the multicast tree. Using NAK as the retransmission request is a scalable solution as shown in [8] and [12]. In this mechanism, when a receiver detects a packet loss, it immediately sends an NAK message to its nearest up stream repair server for retransmission. If the packet is cached there, the repair server sends the packet back to the receiver. Otherwise, the repair server forwards the NAK request to its nearest up stream server for retransmission. This process is repeatedly done until the packet is recovered.

The buffer space of a repair server is likely to be shared by multiple multicast sessions and/or other applications such as Web caching. Usually a repair server can only cache part of the received packets at a time and hence needs a good

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 423–432, 2007.

cache management scheme to cache those packets which have high probability to be retransmitted in the near future. Moreover, due to the hierarchical nature of the repair servers, a cache replacement algorithm should consider the cache cooperation at different levels to achieve good cache performance. Let us consider a 2-level hierarchical cache system with equal sized caches as shown in Fig. 1. Using first-in first-out (FIFO) replacement algorithm [6] [9], each coming packet from the sender is cached in both the level-0 and level-1 caches. The both caches have the same cached packets. Hence, the level-0 cache has no any help in loss recovery, because a lost packet is either recovered in the level-1 cache or missed in the both level-1 and level-0 caches.

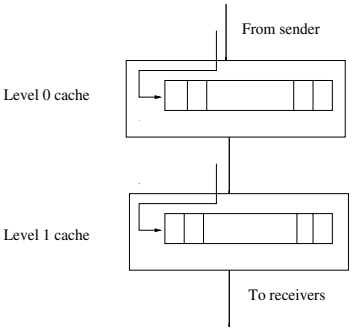


Fig. 1. An example of a two-level hierarchical cache system

Significant efforts have been made on the development of efficient cache replacement algorithms for reliable multicast [4] [6] [13] [14] [15]. All these existing cache replacement algorithms aim to seek the optimized cache performance in a single cache. However, an optimized cache replacement algorithm for a single cache may not achieve the optimized cache performance in a hierarchical cache system. Therefore, a replacement algorithm has to consider cache cooperation to achieve optimized cache performance in a hierarchical cache system.

This paper aims to address the fundamental design problems for hierarchical cache system design in reliable multicast. In this paper, we first derive a cooperative caching efficiency model for a hierarchical cache system. The key difference between our model and existing ones is that the cache efficiency in our model not only depends on the packet caching time, but also on the packet retrieving latency. Based on the efficiency model, a heuristic *C*ooperative *C*ache *R*eplacement (CCR) algorithm is proposed to achieve effective cache performance in reliable multicast. The implementation issues of the CCR algorithm including the up stream cache content estimation and cache management are discussed in details. Extensive simulations are experimented to evaluate the performance of CCR. The simulation results show that CCR can effectively reduce the packet loss recovery latency compared to the OCT algorithm.

The rest of the paper is organized as follows. In section II, we give a brief review of existing cache replacement schemes in reliable multicast. Section III

describes the details of the proposed two-level cooperative hierarchical cache replacement algorithm. The performance comparisons of the proposed scheme with existing OCT are also presented. Finally, Section IV concludes the paper.

2 Related Work

The simplest cache replacement algorithm is First-In First-Out (FIFO) [6] [9]. In FIFO, each coming packet is cached in a repair server. If there is no free space, the oldest packet is dropped. FIFO is still attractive due to its easy implementation. However, when the traffic load is heavy, each packet can only be cached in a short time such that it has no chance to be retransmitted before its dropout. In order to avoid too short caching time, a Probabilistic FIFO (P-FIFO) was proposed [4] [9]. In P-FIFO, when a cache is full, each cached packet has the same probability to be dropped out to release the space for a new coming packet. The drop probability depends on the packet arrival rate, the cache buffer size and the average NAK latency. In order to maximize the caching efficiency, a Timer-Based Caching Policy (TBCP) [6] [9] [13] was proposed. In TBCP, a new coming packet replaces an old one only if the timer of the old packet expires. The key issue in this algorithm is how to select the timer time based on the NAK latency distribution. The experimental results show that the algorithm achieves the best performance by set the timer time to 1.2 round-trip time (RTT). However, there is no theoretical work to explain this observation. Very recently, an Optimal Caching Time (OCT) [14] based cache replacement scheme was proposed to achieve the near optimized cache performance. A caching efficiency model based on the NAK latency distribution is derived and applied to calculate the optimal packet caching time.

OCT achieves the near optimized cache performance in a single cache. However, an optimized cache algorithm designed for a single cache may not lead to an optimized cache performance in a hierarchical cache system. Although some cooperative cache replacement schemes have been developed in hierarchical Web cache systems [3] [11], these schemes cannot be directly applied to reliable multicast system due to the differences of the two systems. Hence it is fundamental and important to develop cooperative cache replacement algorithms to achieve the optimized cache performance in reliable multicast.

3 Cooperative Cache Replacement Algorithm in Two-Level Hierarchical Cache Systems

In this section, we first discuss how to achieve optimal cache performance in a two-level hierarchical cache system, then derive a cooperative caching efficiency model. Based on the model, a heuristic *C*ooperative *C*ache *R*eplacement (CCR) algorithm is proposed to achieve effective cache performance in two-level hierarchical cache systems.

3.1 Optimal Cache Performance in Hierarchical Cache Systems

Let us first evaluate the contributions of cached packets in loss recovery. Consider a two-level hierarchical cache system with a single cache (C_0) of size S_0 at level-0 and M caches (C_k) of sizes S_k (k=1, 2, ..., M) at level-1. The cache system architecture is shown in Fig. 2. Assume the links from the sender to level-1 caches are loss free and the links from the level-1 caches to receivers are lossy. The packet transmission time including propagation and queue delay from the sender to C_0 is d_0, from C_0 to level-1 cache C_k is d_k, and from level-1 cache C_k to receiver j is d_{kj}. Suppose that receiver j connected to repair server C_k loses a packet, i. Without caches, the recovery latency of packet i is

$$R_i^{nc} = 2(d_{kj} + d_k + d_0) \tag{1}$$

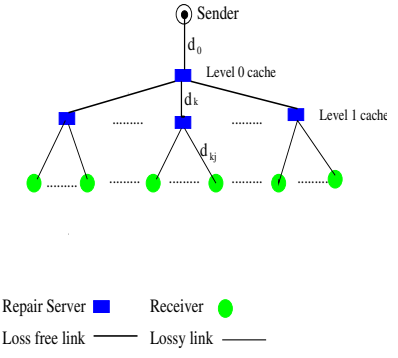


Fig. 2. A general two-level hierarchical cache system architecture

With caches, the recovery latency of packet i is

$$R_i^c = 2[d_{kj} + d_k(1 - \delta_i^k) + d_0(1 - \delta_i^k)(1 - \delta_i^0)] \tag{2}$$

where δ_i^k is 1 if packet i is cached in C_k (k = 0,1,...,M), and 0 otherwise. With caches, the loss recovery latency is reduced by δR_i:

$$\delta R_i = R_i^{nc} - R_i^c = 2[d_k\delta_i^k + d_0(\delta_i^0 + \delta_i^k - \delta_i^0\delta_i^k)] \tag{3}$$

Eqn. 3 shows the contribution of a cached packet in loss recovery. If a packet is only cached in a level-1 or 0 cache, the cached packet in the level-1 cache reduces $2(d_k + d_0)$ loss recovery latency and in the level-0 cache reduces $2d_0$ latency. When a packet is cached in both level-1 and 0 caches, a cached packet in the level-1 cache can be considered only reducing the loss recovery latency by $2d_k$, because a lost packet can be recovered from the level-0 cache if it is not cached in the level-1 cache. While the level-0 cache can be considered as no contribution in loss recovery, because a lost packet can be recovered from the level-1 cache. Hence the contribution of a cached packet depends on cache

content at different levels. A cooperative efficiency model should take this into account.

Let p_{jk} be the packet loss probability of the link from level-1 cache C_k to receiver j; $F^0_{kj}(t)$ represents the cumulative probability that the NAK latency at C_0 coming from receiver j through the level-1 cache C_k is smaller than or equal to t time, and $F^k_j(t)$ represents the cumulative probability that the NAK latency at level-1 cache C_k coming from receiver j is smaller than or equal to t time. Assume packet i has been cached in C_k for t^k_i time and the optimal caching time for packet i is t^o_{ki}. Then caching packet i has probability $p_{jk}[F^k_j(t^o_{ki}) - F^k_j(t^k_i)]/(t^o_{ki} - t^k_i)$ and $p_{jk}[F^0_{jk}(t^o_{0i}) - F^0_{jk}(t^0_i)]/(t^o_{0i} - t^0_i)$ to be retransmitted by receiver j per unit time in level-1 cache C_k and C_0, respectively. Therefore the total reduced loss recovery latency per unit time (δR) due to the cache system is

$$\delta R = \sum_{k=1}^{M} \sum_{j} \sum_{i} 2p_{jk}[\frac{F^k_j(t^o_{ki}) - F^k_j(t^k_i)}{t^o_{ki} - t^k_i}\delta^k_i d_k + \frac{F^0_{jk}(t^o_{0i}) - F^0_{jk}(t^0_i)}{t^o_{0i} - t^0_i}(\delta^0_i + \delta^k_i - \delta^0_i\delta^k_i)d_0]$$

The optimal cache performance of a cooperative cache system is to seek the minimum loss recovery latency, that is to maximize the reduced loss recovery latency, i.e.,

$$Maximize \qquad \delta R \tag{4}$$

$$Subject\ \ to \qquad n_k \leq S_k, \quad k = 0, 1, .., M. \tag{5}$$

where n_k is the number of cached packets in C_k.

The optimal cache replacement algorithm maximizes the reduced packet loss recovery latency, δR. To compute δR, each cache needs to know the cache information including the cache content as well as the NAK latency distribution of each other cache at different levels. When a new packet arrives at C_0, C_0 needs to compute δR by assuming: (1) the packet is cached in C_0; and (2) the packet is not cached in C_0. In each case, C_0 first determines which level-1 caches will cache the packet, and computes δR. Then the values of δR in two cases are compared. The packet is cached by C_0 if and only if δR in the first case is larger than that in the second case. When a level-1 cache receives the packet, it first needs to know if the packet is cached in C_0 or not, then decides to cache the packet or not by computing δR. However, frequently exchanging cache information between C_0 and level-1 caches is too costly. Moreover, the computation cost is also high in C_0 for computing the caching efficiency for every level-1 cache. In the following, we design a heuristic Cooperative Cache Replacement (CCR) algorithm to achieve near optimized cache performance in a two-level hierarchical cache system with minimized communication overhead.

3.2 Heuristic Cooperative Cache Replacement (CCR) Algorithm

A heuristic *C*ooperative *C*ache *R*eplacement (CCR) algorithm is proposed to seek the near optimized cache performance in a two-level hierarchical cache system with minimized communication cost. In CCR, the replacement algorithm of C_0 is an optimized single cache replacement algorithm, while a level-1 cache seeks the optimized cache performance by taking into account the cooperation with C_0. In C_0, the caching efficiency is computed based on the cumulative distribution of NAK latency [14]. The caching efficiency$U^0(\tau_i, t)$ of packet i already cached by τ_i time is defined as

$$U^0(\tau_i, t) = \sum_k \sum_j p_{kj} \frac{F^0_{kj}(t) - F^0_{kj}(\tau_i)}{t - \tau_i} \tag{6}$$

Here the summation is over all possible receiver j connected through all possible level-1 cache C_k. A new coming packet replaces an old one only if it has higher caching efficiency than that of the old one. The caching efficiency of a dropout packet can be modeled by an optimal dropout time t^0_o which corresponds to a value of t with maximum $U^0(0, t)$ [14]. Therefore, only these caching packets which have caching time longer than t^0_o have less caching efficiency than that of a new coming packet. Hence, a new coming packet is cached by replacing an old one only if the oldest one is cached longer than t^0_o.

In level-1 cache C_k, the replacement algorithm minimizes the loss recovery latency by taking into account the cooperation with C_0. The contribution of a cached packet in C_k not only depends on its caching time, but also on its retrieving latency. If a packet is not cached in C_0, C_k takes $2(d_k + d_0)$ time to retrieve it from the sender. Otherwise, C_k only takes $2d_k$ time to recover it from C_0. Based on these observations, the caching efficiency of a packet which is cached in C_0 is reduced by a factor of $d_k/(d_k + d_0)$. Now the cooperative caching efficiency$U^k(\tau_i, t)$ for packet i which has been cached for τ_i time in C_k $(k \neq 0)$ is computed as

$$U^k(\tau_i, t) = \sum_j p_{kj} \frac{F^k_j(t) - F^k_j(\tau_i)}{t - \tau_i} \times \frac{d_k + (1 - \delta^0_i)d_0}{d_k + d_0} \tag{7}$$

In Eqn. (7), the caching efficiency of a cached packet depends not only on the cumulative distribution of NAK latency, but also on the cooperation with C_0 (i.e., δ^0_i). The caching efficiency of a packet cached in C_0 is reduced by a factor of $d_k/(d_k + d_0)$. When a new packet arrives, C_k first checks if the packet is cached in C_0 or not, then calculates its caching efficiency. If no cached packet has less caching efficiency than that of the new one, the new one is dropped. Otherwise, the old one with minimal caching efficiency is replaced by the new packet. The caching efficiency calculation in Eqn.(7) takes the cache cooperation into account and hence improves the cache performance in hierarchical cache systems. In the follows, we describe the implementation issues of CCR.

3.3 Implementation of Cooperative Cache Replacement Algorithm

In this subsection, we address three important implementation issues of CCR: (i) how a level-1 cache knows the cache content of C_0; and (ii) how a level-1 cache manages its cache content.

We propose a solution for a level-1 cache to estimate the cache content of C_0 here. In our scheme, C_0 passes its dropout time t_o^0 and its cache size S_0 to every level-1 cache, C_k. C_k maintains the information (packet sequence number and arrival time) of every cached packet in C_0. Initially, C_k knows that the first S_0 packets are cached in C_0. Then a coming packet is cached or not by comparing the caching time of the oldest packet with t_o^0. Due to each packet takes d_k time from C_0 to C_k, C_k can use the packet arrival time at itself instead of knowing the packet arrival time at C_0 to make the decision.

In level-1 cache C_k, all caching packets are stored in two lists: one for the packets which are not cached in C_0, the other for the packets which are cached in C_0. Each list is a sorted list according to the packet arrival time, the newest packet is at head. The caching efficiency of a packet depends on its caching time. As shown in [14], there is a critical caching time, when the caching time of a packet is less than the critical value, the caching efficiency increases as caching time increases. But if the caching time is over the critical value, the caching efficiency decreases as caching time increases. So the packets at the list tail or head usually have the lowest caching efficiency. Hence a new coming packet only needs to compare the caching efficiency with at most four packets. When a new packet arrives, C_k first estimates if it is cached in C_0. The caching efficiency of the new packet is calculated by Eqn. (7). Then its caching efficiency is compared to that of four caching packets (head and tail packets in two lists). If it has higher efficiency than any of the four packets, the cached packet with lowest efficiency is dropped from the cache and the new packet is added to the corresponding cache list head. Otherwise, the new packet is dropped.

3.4 Performance Evaluation

We use Network Simulator 2 (ns-2) [10] to study the performance of the proposed CCR algorithm in two-level hierarchical cache systems. OCT is used for performance comparison because it has the best cache performance among all the single cache replacement algorithms as shown in [14]. We use cache hit ratio and loss recovery latency as two performance metrics. The cache hit ratio is defined as the number of recovered NAKs divided by the total number of received NAKs in a repair server. The loss recovery latency is defined as the time interval between a receiver first detecting the packet loss and the receiver getting the recovered packet. The Pragmatic General Multicast (PGM) [5] is used as the reliable multicast protocol. NAK is used for packet retransmission, and a repair server may send the suppressed NAKs to its upstream nodes. In the simulations, the extra cache space for storing packet sequence number and arrival time is ignored in both OCT and CCR because the both algorithms need the same storage space.

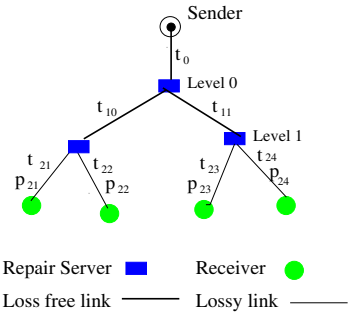


Fig. 3. Two-level network topology

We study the cache performance in a heterogenous network. The network topology is shown in Figure 3. The system parameters are set as: $t_0 = 80\ ms$, $t_{10} = 20\ ms$, $t_{11} = 40\ ms$, $t_{21} = 20\ ms$, $t_{22} = 80\ ms$, $t_{23} = 30\ ms$, $t_{24} = 60\ ms$; $P_{21} = 0.005$, $P_{22} = 0.02$, $P_{23} = 0.01$, $P_{24} = 0.03$. The bandwidth for each link is set to 2 Mbps. The traffic is a Poisson traffic with rate 672 Kbps and the packet size is set as 226 Bytes.

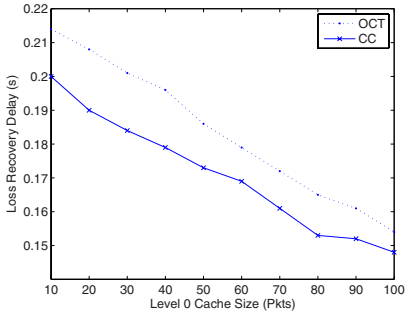


Fig. 4. Loss recovery latency, heterogenous network, $C_k = 40$

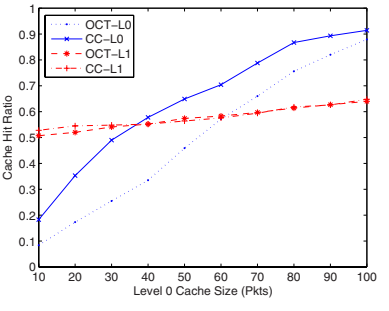


Fig. 5. Cache hit ratio, heterogenous network, $C_k = 40$

Figs. (4) - (7) show the loss recovery latency and cache hit ratio of CCR and OCT at level-1 cache size of 40 and 50, respectively. From these figures, we observe the similar results as those in the homogenous case. This indicates that CCR works well in heterogenous networks. One interested thing is that the level-1 cache hit ratio increases as the size of C_0 increases. When the size of C_0 increases, the cache hit ratio of C_0 increases, and hence the number of NAKs reaching the sender is reduced. This results in less traffic load and shorter queue delay in the sender. In this case, the receivers can sense packet loss earlier, and hence the average NAK latency in level-1 caches is reduced. Therefore cache hit ratio is increased due to the short effective packet caching time (the dropout time). When the size of level-1 caches increases from 40 to 50, the cache hit

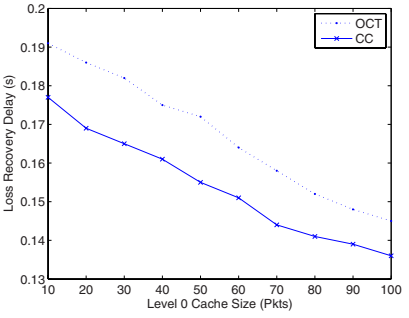


Fig. 6. Loss recovery latency, heterogenous network, $C_k = 50$

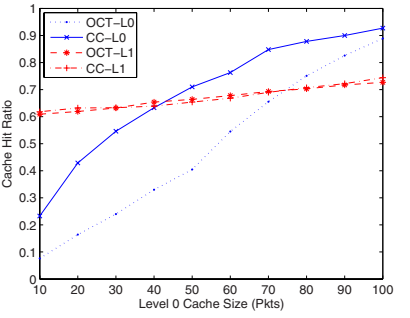


Fig. 7. Cache hit ratio, heterogenous network, $C_k = 50$

ratio of level-1 caches increases and hence the loss recovery latency is reduced in both CCR and OCT. The cache hit ratio in C_0 is almost not changed. The performances of both CCR and OCT are similar in two different level-1 cache sizes.

The simulation results show that the proposed CCR algorithm is superior to the optimized single cache algorithm OCT in both homogenous and heterogenous networks.

4 Conclusions

Packet loss recovery plays a critical role in reliable multicast. One effective way for packet loss recovery is to place repair servers with active routers along the transmission paths. Usually a repair server can only cache part of the received packets at a time, hence a good cache management policy is necessary to improve the cache performance. The existing cache management policies in reliable multicast were designed to achieve optimized cache performance in a single cache. However, the repair servers in reliable multicast is a hierarchical cache system, an optimized cache replacement designed for a single cache may not lead to optimized cache performance in the whole system. Therefore it is fundamental important to design a hierarchical cooperative cache system to achieve optimized cache performance in the whole system.

In this paper, we proposed a cooperative cache replacement (CCR) algorithm for hierarchical caching system to achieve near optimized cache performance in reliable multicast. The key difference between our proposed cache replacement algorithm and the existing ones is the efficiency model in which the caching efficiency of a packet not only depends on its caching time but also on the packet retrieving latency. The implementation details of the CCR algorithm in two-level hierarchical cache systems are discussed. Various simulations are conducted in both homogenous and heterogenous networks. The results show that CCR effectively reduces the loss recovery latency and significantly increases the cache hit ratio in low level caches.

Acknowledgments. The authors would like to express their gratitude to Dr. Feng Xie and Dr. Gang Feng from Nanyang Technological University for kindly providing their OCT simulation codes with us.

References

1. Cao, G.: Proactive Power-aware Cache Management for Mobile Computing System. IEEE Transactions on Computers 51(6), 608–621 (2002)
2. Cao, P., Irani, S.: Cost-aware WWW Proxy Caching Algorithms. In: Proceedings of USENIX Symp, Internet Technology and Systems, pp. 193–206 (1997)
3. Che, H., Tung, Y., Wang, Z.: Hierarchical Web Caching Systems: Modeling, Design and Experimental Results. IEEE Journal of Selected Areas in Communications 20(7), 1305–1314 (2002)
4. Feng, G., Yeung, K.L., Kheong, S.C.: Optimal Cache Allocation and Probabilistic Caching for Local Recovery in Reliable Multicast. In: Proceedings of IEEE ICC, IEEE Computer Society Press, Los Alamitos (2000)
5. Gemmell, J., Montgomery, T., Speakman, T., Crowcroft, J.: The PGM Reliable Multicast Protocol. IEEE Networks 17(1), 16–22 (2003)
6. Kasera, S., Kurose, J., Towsley, D.: Buffer Requirements and Replacement policies for Multicast Repair Service. In: Proceedings of Second International Workshop on Networked Group Communication (2000)
7. Kasera, S., Bhattacharyya, S., Keaton, M., Kiwior, D., Zablele, S., Kurose, J., Towsley, D.: Scalable Fair Reliable Multicast Using Active Services. IEEE Network 14(1), 48–57 (2000)
8. Lehman, L., Garland, S., Tennenhouse, D.: Active Reliable Networks. IEEE INFOCOM (1998)
9. Leung, K.L., Wong, H.T.: Caching Policy Design and Cache Allocation in Active Reliable Multicast. Computer Networks 43(2), 177–193 (2003)
10. ns-Network Simulator, http://www.isi.edu/nsnam/ns
11. Rizzo, L., Vicisano, L.: Replacement Policies for a Proxy Cache. IEEE Transactions on Networks, 8(3), 158-170 (2000)
12. Towsley, D., Pingali, S.: A Compairson of Sender-Initiated and receiver-initiated Reliable Multicast Protocols. IEEE Journal of Selected Areas of Communications 15(3), 398–406 (1997)
13. Zhen, X., Birman, K., Renesse, R.: Optimizing Buffer management for Reliable Multicast. In: Proceedings of International Conference on Dependable Systems and Networks (DSN) (2002)
14. Xie, F., Feng, G., Yang, X.: Optimizing Cache Policy for Loss Recovery in Reliable Multicast. In: Proceedings of IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2006)
15. Yeung, K., Wong, H.: Caching Policy Design and Cache Allocation in Active Reliable Multicast. Computer Networks 43(2), 177–193 (2003)

A Novel Design of Hidden Web Crawler Using Reinforcement Learning Based Agents

J. Akilandeswari[1] and N.P. Gopalan[2]

[1] Department of Computer Science and Engineering, Sona College of Technology, Salem, Tamil Nadu, India
akila_rangabashyam@yahoo.com
[2] Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
gopalan@nitt.edu

Abstract. An ever-increasing amount of information on the Web today is available only through search interfaces: the users have to type in a set of keywords in a search form in order to access the pages from certain Web sites. These pages are often referred to as the *Hidden Web* or the *Deep Web*. Since there are no static links to the Hidden Web pages, search engines cannot discover and index such pages and thus do not return them in the results. However, according to recent studies, the content provided by many Hidden Web sites is often of very high quality and can be extremely valuable to many users. In this paper, an effective design of Hidden Web crawler *ALAC* that can autonomously discover pages from the Hidden Web is discussed. Here, a theoretical framework is presented to investigate the resource discovery problem. This article proposes an effective crawling strategy for identifying hidden web sites automatically. The crawler design employs agents fuelled with reinforcement learning. The prototype is experimentally evaluated for the effectiveness of the strategy and the results are very promising. The crawler *ALAC* has found 567 searchable forms after searching 3450 pages which substantiate the effectiveness of the policy.

Keywords: Web Crawler, Hidden Web, Intelligent Agent, Reinforcement Learning, Web mining.

1 Introduction

Recent research shows that though the *surface web* has linked billions of static web pages, a part of Web content cannot be reached by following hyperlinks [2, 4]. A large part of the Web is available behind search interfaces and is reachable only when users fill up those forms with set of keywords or queries. These pages are often referred to as the *Hidden Web* [3] or *Deep Web* [2]. Search engines' crawlers typically cover only Publicly Indexable Web (PIW). A deep web site is a web server that provides information maintained in one or more back- interfaces [4]. According to many research studies, the size of the Hidden Web rapidly increases as more organizations put their valuable content online through an easy-to-use Web interface [2] and is now estimated as 550 times larger than the surface web [1]. The content provided by

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 433–440, 2007.

many Hidden-Web sites is and web databases, each of which is searchable through one or more HTML forms. The information from these sites is of very high quality and can be extremely valuable to many users [2]. For example, PubMed provides many high-quality papers on medical research which can be of helpful to medical practioners and medical researchers.

Given this dynamic nature of web with new sources constantly being uploaded and old information are either removed or modified, it has become very important to automatically discover hidden web sites through an easy to use interface. The traditional web crawler automatically traverse, retrieve pages and build a large repository of web pages. The application of hidden web crawler differs from the traditional crawler. The usage of the former is for obtaining web databases that are of high quality and many researches or organizations may use the database for integration. To corroborate, a breadth first crawler indexes only 23 searchable forms in bioinformatics domain after crawling 10000 pages. This scenario urges to develop a crawler that navigates through various web sites which should have the following properties:

1. automatically find the hidden resources
2. should accurately specify a schema that describes the relevant forms which is very hard even in a well formed domain
3. need to perform a broad search
4. search process must be efficient and avoid visiting large unproductive portions of the web
5. must produce high quality results

This paper discusses the design of such a web crawler for indexing hidden web pages or databases in a particular domain. The whole problem has two tasks: resource discovery and content extraction [5]. The first task deals with automatically finding the relevant web sites containing the hidden information. The second task deals with obtaining the information from those sites by filling out forms with relevant keywords. The proposed work describes on how to locate relevant forms that serve as the entry points to the hidden web data.

The crawler design has two approaches: exhaustive crawling and topic specific crawling [6]. The first approach is time consuming and gives low harvest rate. The second approach can be the best approach since the users who access the hidden web will be searching based on some particular topic. Focused crawlers have increased crawling efficiency in terms of quality indices [7]. In [6,8] an adaptive form focused crawler is built that identifies the relevant web resources. In that work, different classifiers are used to identify the topic specificity, to determine the promising links to follow, to discriminate between searchable and non-searchable forms. The use of these classifiers at different stages may consume more time.

Our paper focuses on designing a crawler using learning agents whose design goal will be to find the hidden resources automatically producing high quality results. The crawler's search process must be efficient and circumvent visiting large unproductive portions of the web. That is, the crawler avoid visiting

i. web pages that are not forms that lead to databases
ii. web forms that are not relevant to the topic of query
iii. web pages that are spam
iv. web forms that are duplicated in mirror sites

This paper discusses the new architecture designed for efficient discovery of the hidden web resources. The results show that the crawler *ALAC* is performing better with respect to harvest rate. The result also indicates that the automatic resource discovery improves as the agent learns progressively.

The remainder of this paper is organized as follows: section 2 discusses the related work. In section 3, the architectural framework of *ALAC* is described. In section 4, the experimentation and evaluation details are discussed. Finally, in section 5, future directions and conclusions are detailed.

2 Related Work

The pioneering work on hidden web crawler design [5] focused on extracting content from searchable electronic databases. They introduced an operational model of HiWE(Hidden Web crawler). In the paper, they have discussed different directions on research in designing the crawler for content extraction. There are works on focused crawlers [7,9,10] which describe the resource discovery on particular domain or topic by PIW crawlers. In [4], useful observations and implications are discussed which are very much helpful for researchers. Their survey reports on locating entry points to hidden web, coverage of deep web directories and so on. They also give a clear observation that the crawler strategy for deep web are likely to be different from surface web crawlers. In [6], form focused crawler for hidden web is described. In [11] an adaptive strategy was employed in designing the classifiers. The proposed work explores the use of learning agents which exploits parallelism and improves the harvest rate.

3 Architectural Framework

Form distribution in a particular domain is of sparse nature [6]. This nature made the crawler to crawl only in the particular domain and thus avoids visiting unproductive paths by learning feature of links. The simplified architecture of the crawler is given in Fig.1.

The learner is implemented as a learning agent, which does two tasks. One is to identify the pages as belonging to particular domain. Another task is to classify the

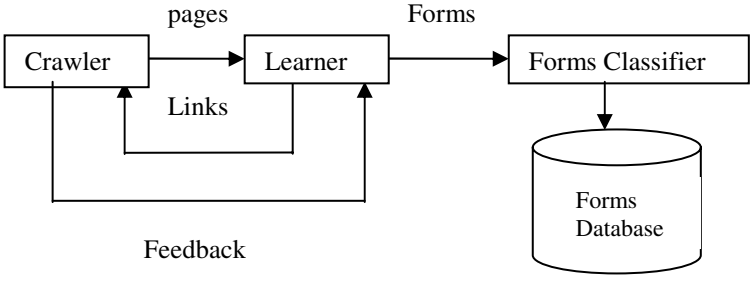


Fig. 1. Simplified Architecture

links which gives immediate benefit and links which give delayed benefit. Once the learner agent identifies the relevant forms in the visited pages, the form classifier again classifies the search interfaces as searchable forms and non searchable forms. This form classification module is important because it is unnecessary to explore irrelevant forms such as login forms, and registration forms. The relevant forms are then stored in the forms database for further investigations. The learner gives progressive feedback to crawler which helps in formulating the crawling policy.

3.1 Learner

The learning module has two sub modules: page classifier and link classifier.

Page classifier. The pages are classified according to [8]. The page classifier is trained using reinforcement learning algorithm. The authors of [12] have proved that the reinforcement learning based agents provide good results in terms of harvest rate and freshness. The purpose of this classifier is to avoid investigating unproductive pages. When the pages are classified as relevant, links in the pages are examined.

Link Classifier. The links are evaluated for its relevance [12] in the document using eqn [1].

$$\textbf{l\_rel(d,p)} = \frac{\sum_{k \in p \cap d} f_{kd} \; f_{kp}}{\min \left(\sum_{k \in d} f_{kd}^{\;2} \right) \left(\sum_{k \in p} f_{kp}^{\;2} \right)} \tag{1}$$

where d is domain based keywords, p is the page under investigation, and k is the frequency of search terms in the feature space.

The feature space considered for computation of link relevance is URL, anchor text, text around the hyperlink. The frequency of the search terms in the feature space is computed. While considering the URL as feature space, the frequency is computed as follows: if any of the search terms are found as substring in the URL, then the frequency of that search term is incremented. The computation of link relevance described above will identify the pages with immediate benefit. That is, following the link will provide the search interface belonging to that particular domain. The objective of the link classifier is also to identify the links which give delayed benefit i.e. the links that eventually lead to searchable interfaces. Learning component of the classifier is involved in classifying the link with delayed benefit.

As in page classifier, the learning component employed in link classifier is reinforcement learning. The learner learns using the same feature set described above and assigns an integer value as score which is the normalized distance of the relevant form from the link to be followed. The learner in this prototype does not need any training sets. It learns from the progressive crawling and modifies its current state using the feedback given from the crawler module as depicted in the Fig. 1. The classifier then gives the links to the crawler, which then prioritizes immediate benefited links over delayed benefited links. Priority queue is implemented for prioritizing.

Forms Classifier. This module identifies the located forms as searchable and non searchable forms. Non searchable forms are the interfaces for login, subscription, and registration. This module uses decision tree classifier which is proved to have lowest error rate [6]. The features considered for classifying the forms are: checkboxes, password tags, submit buttons, textboxes, number of items in select box, submission method and search keyword in the form [6].

3.2 Learning Methodology

The adaptability of the agent in the learner is incorporated through reinforcement learning. The relevance of the link to be followed is computed based on the above mentioned methodology. The term reinforcement learning refers to a framework for learning optimal decision making form rewards or punishment. It differs from the supervised learning in that the learner is never told the correct action for a particular state, but simply gives the feedback as how good or bad the selected action was, expressed in terms of scalar reward [13]. A task is defined by a set of states which are the set of already retrieved documents s ε S, a set of actions which are the documents to be retrieved a ε A, a state-action transition function T : S X A → S and a reward function R : S X A → R. The goal of reinforcement learning is to learn a policy, a mapping from states to actions, Λ : S → A, that maximizes the sum of its reward over time. The infinite-horizon discounted model is chosen to compute the long-term reward for a given sequence of rewards from separate steps.

$$U\ (\Lambda) = E_{\Lambda}\left(\sum_{k=1}^{\infty} \gamma^{k-1} r_k\right) \tag{2}$$

where Λ is policy and k is the steps. The evaluation is optimal if it yields the maximum possible expected long-term reward. Let Q* (s,a) be the value of selecting a link a from set of states s and thereafter following the optimal policy. The state space is the set of links in the document under consideration.

$$Q^*(s,a) = R(s,a) - \gamma V^*(T(s,a)) \tag{3}$$

where V* is the value of each link and is computed as

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{4}$$

and r_t is the reward received t time steps after starting in state s and following policy π. The optimal policy can be defined in terms of Q by selecting from each state the action with the highest expected future reward π* (s) = argmax Q*(s,a). The Q function is a mapping from selecting from an action to a scalar value.

4 Experimentation and Evaluation

The web crawler design exploits computational parallelism to achieve efficiency through multithreading programming. The agent framework is implemented using

java based aglets platform. The parameters are kept transparent from the users to keep the search interface as simple as possible. The input to the search interface is the keywords which describe the domain, and a starting URL to initiate the crawler. To compute the energy given to the agent, latency is considered which is the time spent on learner module.

$$\text{Latency} = \frac{time(p)}{MAX} * \frac{time(l)}{TIMEOUT} \tag{5}$$

where time(p) is the time taken to download the page by the crawler, time(l) is the time taken in the learner module to output the links to the crawler, MAX is the limit set on total number of pages to be crawled, and TIMEOUT is the expiry time of the socket connection.

The system is assessed and compared with two other crawler designs:

- Topic specific crawler A (TSC A): the crawler follows all the links from the page which is classified as being on topic.
- Topic specific crawler B (TSC B): crawler that follows links only to specific levels of depth. In the experiment, the depth is set as 2.

The new crawler design is evaluated based on number of relevant forms harvested, time taken to download forms, and learning effectiveness. Three domains are considered for testing: hotels, music and bioinformatics whose data can be retrieved by filling out forms.

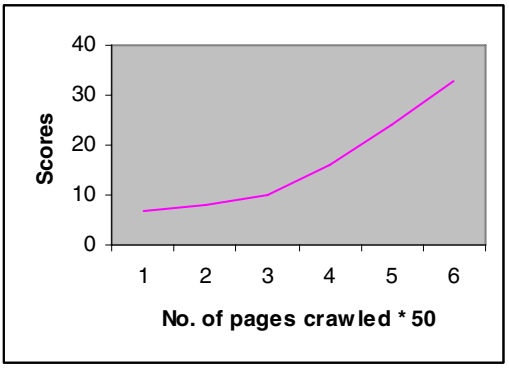


Fig. 2. Learning Effectiveness

The fig. 2 above shows the learning effectiveness of the learner module. The above graph clearly depicts that at early stages the crawler is ineffective, but after gaining learning experience, the reward or the score computed is satisfactory. As the crawler visits more number of pages the scores are maximized accordingly.

Fig. 3 shows the behavior of the crawler based on the number of relevant forms retrieved. During the initial crawls all the three schemes are performing equally. After gaining the learning strategy ALAC is outperforming the other two schemes. Table 1 shown below depicts the number of relevant forms retrieved after traversing 10000 pages in the domains: hotels, music and bioinformatics.

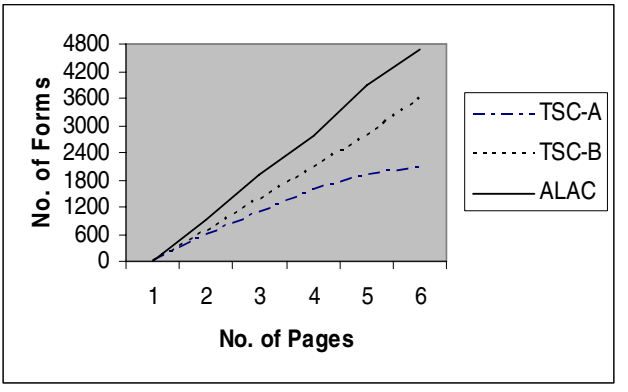


Fig. 3. Number of relevant forms retrieved Vs Total number of forms visited

Table 1. Number of relevant forms retrieved

	Hotels	Music	Bioinformatics
TSC-A	159	127	236
TSC-B	675	348	543
ALAC	1038	769	1350

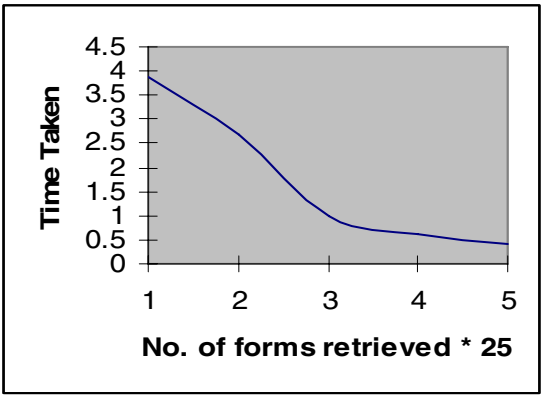


Fig. 4. Performance based on time taken to retrieve forms

In fig. 4, the performance of the crawler based on time taken to retrieve the forms is depicted. In the early stages, due to the slow learning rate, the time consumed to retrieve first 100 forms is more. Later it is improvising over the time. Our crawler has taken 2.79 minutes to retrieve 91 forms in bioinformatics domain and later it took 0.56 seconds to retrieve 100 forms. The above results show the effectiveness of the crawler design based on the factors discussed above.

5 Conclusion and Future Direction

The explosive nature of Web needs a crawler which can identify Web pages that are of high quality to the users. This article proposed a new architectural framework for hidden web crawler *ALAC*, which automatically identifies the hidden resources. The framework suggested uses learning agents mining the documents online thus reducing the time taken for retrieval. The evaluation results show that the crawler is able to perform efficiently by focusing the search in particular domain. It is able to learn effectively which indeed reduces the time consumed in retrieving searchable forms. The crawler design is compared with two other traditional crawler designs and the results are promising in terms of harvest rate.

Actually the design can be extended to incorporate multiple agents in the learner and crawler to exploit the parallelism. As an extension the design can be incorporated to employ intelligent extraction framework to automatically dig out the contents or the databases of the deep web resource.

References

1. BrightPlanet. Com, The deep web: Surfacing hidden value (July 2000), http://brightplanet.com
2. Bergman, M.K.: The deep web: Surfacing the hidden value, http://www.press.mich.edu/jep/07-01/bergman.html
3. Florescu, D., Levy, A.Y., Mendelzon, A.O.: Database techniques for world wide web: A Survey. SIGMOD record 27(3), 59–74 (1998)
4. Chang, K.C.-C., He, B., Li, C., Patel, M., Zhang, Z.: Structured databases on the Web: Observations and Implications, Technical Report, UIUC
5. Raghavan, S., Garcia-Molina, H.: Crawling the Hidden Web. In: Proc. of the 27th VLDB Conference (2001)
6. Barbosa, L., Freire, J.: Searching for hidden-web databases. In: Eighth Intl. workshop on the Web and Databases (2005)
7. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: A New Approach to Topic specific Web Resource Discovery. Computer Networks 31(11-16), 1623–1640 (1999)
8. Akilandeswari, J., Gopalan, N.P.: A Web Mining System using Reinforcement Learning for Scalable Web Search with Distributed. Fault-tolerant Multi-agents, WSEAS transactions on Computers 4(11), 1633–1639 (2005)
9. Diligenti, M., Coetzee, F., Lawrence, S., Giles, C.L., Gori, M.: Focussed Crawling using Context Graphs. In: Proc. of the 26th Intl conf. on Very Large Databases, pp. 527–534 (2000)
10. Miller, R.C., Bharat, K.: Sphinx: A framework for creating personal, site specific web crawlers. In: Proc. for the 7th Intl WWW conf. (1998)
11. Barbosa, L., Freire, J.: An Adaptive Crawler for Locating Hidden-Web Entry Points. In: Proc. of Intl WWW conf., pp. 441–450 (2007)
12. Rennie, J., McCallum, A.K.: Using Reinforcement Learning to Spider the Web Efficiently. In: Proc. of 16th Intl. conf. on Machine Learning (1999)
13. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 237–285 (1995)

Look-Ahead Adaptive Routing on *k*-Ary *n*-Trees

Quanbao Sun, Liquan Xiao, and Minxuan Zhang

PDL, School of Computer, National University of Defense Technology, China
alpha_slt@163.com

Abstract. Supporting multicast at hardware level is a future trend of interconnection networks, and the latency of hardware-based multicast is sensitive to network conflict. In this paper, we study the method to forecast and reduce the conflict between unicast and multicast traffic on *k*-ary *n*-trees. We first derive a switch grouping method to describe the relationship among switches being passed through by a unicast routing path or a multicast tree. Then we analyze the sufficient condition for conflict-free routing. Based on these observations, a look-ahead adaptive routing strategy for unicast packet is proposed. The simulation results indicate that the proposed strategy can lower the multicast latency and the unicast latency simultaneously.

1 Introduction

A high bandwidth and low latency network is essential for high performance parallel system. The *k*-ary *n*-tree [1] is a kind of bidirectional multistage interconnection networks which provides high degree of connectivity leading to high bandwidth and has being used in some of the most powerful computers in the world [2].

Multicast is an important operation in multicomputer communication systems and can be used to support several other collective communication operations [3]. The unicast based approach has higher latency as it does not exploit the concept of minimizing the traffic by multicasting. Supporting multicast at hardware level has been suggested to further enhance the performance, which also is a future trend of interconnection networks [4][5]. The *k*-ary *n*-tree inherits the tree structure which can be used to efficiently support multicast operation, and several hardware-based multicasting schemes on it have been proposed [6][7][8]. The previous researches on hardware-based multicast only consider the routing of multicast packet.

Multicast latency is more sensitive to network conflict than unicast latency. Our previous work has proposed two parent selecting strategies which can be used in multicast tree building algorithm to reduce the conflict among multicast packets [9]. When both unicast and multicast traffic is present, the strategy which can reduce the conflict between these two kinds of traffic is also very attractive. The minimal routing of unicast packet between a pair of processing nodes on a *k*-ary *n*-tree experiences two phases: an ascending phase followed by a descending phase, and only in the ascending phase the packet can use multiple output port. But from the operation characteristic of tree-based multicast we can see that the conflict related to multicast occurs mostly in the descending phase. Therefore, we need a method, which can forecast and reduce the conflict (occurs in the descending phase) between unicast and

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 441–449, 2007.

multicast traffic, to guide the output port selecting for ascending unicast packet. This paper focuses on the solution of this problem. We derive a switch grouping method to describe the relationship among switches being passed through by a unicast routing path or a multicast tree on *k*-ary *n*-trees and analyze the sufficient condition for conflict-free routing. Based on these observations, we propose a look-ahead adaptive routing strategy for unicast packet. At last, the proposed routing strategy is evaluated by a synthetic workload driven simulator.

The rest of this paper is organized as follows: Section 2 will introduce the definition of *k*-ary *n*-trees and the routing scheme on it. The switch grouping method and the proposed look-ahead adaptive routing algorithm will be described in section 3. Section 4 will give the simulation model and simulation results. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 *k*-Ary *n*-Trees

A *k*-ary *n*-tree is composed of two types of vertices: k^n processing nodes and nk^{n-1} $k*k$ communication switches. Each processing node is an *n*-tuple $\{0,1,\cdots,k\text{-}1\}^n$, while each switch is defined as an ordered pair $<\omega,l>$, where $\omega\in\{0,1,\cdots,k\text{-}1\}^{n-1}$ and $l\in\{0,1,\cdots,n\text{-}1\}$. Two switches, $<\omega_0,\omega_1,\cdots,\omega_{n-2},l>$ and $<\omega_0',\omega_1',\cdots,\omega_{n-2}',l'>$, are connected by an edge if and only if $l'=l+1$ and $\omega_i=\omega_i'$ for all $i\neq l$. There is an edge between the switch $<\omega_0,\omega_1,\cdots,\omega_{n-2},n\text{-}1>$ and the processing node $<p_0,p_1,\cdots,p_{n-1}>$ if and only if $\omega_i=p_i$ for $i\in\{0,1,\cdots,n\text{-}2\}$ [1]. An example of a 4-ary 3-tree is shown in Fig 1.

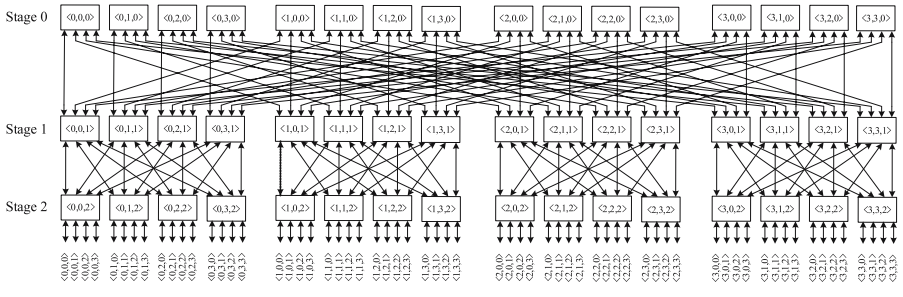


Fig. 1. An example of 4-ary 3-tree

2.2 Routing Scheme

Unicast. Minimal routing between a pair of processing nodes on a k-ary n-tree can be slimly accomplished by sending the packet to one of the nearest common ancestors of both source and destination, and from there following a unique downward path to the destination. That is, each packet experiences two phases, an ascending phase followed by a descending phase.

The goal of adaptive routing on a *k*-ary *n*-tree is to achieve load balance across the different physical links in the ascending phase. The ascending phase can be viewed as an output port allocation problem since the packet can traverse using any uplinks. The full adaptive routing strategy proposed in [10] allows unicast packet using any idle output port in the ascending phase.

Multicast. To deliver a packet to a number of destination nodes, the packet header must carries the destination set information which is used by switches to make appropriate routing decision. The concept of multi-destination packet passing and multi-port encoding has been proposed for implementing fast hardware-based multicast [11]. These two strategies lead to a long packet header as the number of processing nodes getting larger, which are not suit for large-scale systems. Using forwarding table with a bit vector of destination ports for each multicast address is another way to implement multicast packet routing [8], which is more attractive because it is suit for large-scale systems. For multicast operation, a packet is sent to all destination ports except the port on which the packet arrives. In this paper we use forwarding table to route the multicast packet.

3 Look-Ahead Adaptive Routing

From the routing scheme on *k*-ary *n*-trees we can see that the conflict related to multicast occurs mostly in the descending phase, but only in the ascending phase a unicast packet can use multiple output port. In this section, we introduce the look ahead adaptive routing algorithm for unicast packet based on the switch grouping method.

3.1 Grouping the Switches

A conflict occurs when two packets arrive at the same switch and request the same output port simultaneously. To forecast and avoid the conflict, the relationship among the switches which are passed through by a unicast routing path or a multicast tree should be found firstly. The following definitions and theorem are needed to describe this relationship.

Definition 1: Given an *k*-ary *n*-tree $FT(k, n)$, for two switches at the same stage $SW=<\omega_0,\omega_1,\ldots,\omega_{n-2}, l>$ and $SW'=<\omega_0',\omega_1',\ldots,\omega_{n-2}', l>$, $gcs(SW,SW')=\omega_\alpha,\ldots,\omega_{n-2}$ is the greatest common suffix of *SW* and *SW'* if $\omega_\alpha,\ldots,\omega_{n-2}=\omega_\alpha',\ldots,\omega_{n-2}'$ and $\omega_{\alpha-1}\neq\omega_{\alpha-1}'$. The length of $gcs(SW,SW')$ is n-1-α. If $\alpha>n$-2, it denotes that the labels of two switches have no common suffix.

Definition 2: We define a relation *R* as follows: Given an *k*-ary *n*-tree $FT(k, n)$, for two switches at the same stage $SW=<\omega_0,\omega_1,\ldots,\omega_{n-2},l>$ and $SW'=<\omega_0',\omega_1',\ldots,\omega_{n-2}',l>$, we say *SW R SW'* if the length of $gcs(SW,SW')$ (recorded as $|gcs(SW,SW')|$) is n-1-l.

From definition 1 and definition 2 we can see that the relation *R* is an equivalence relation. According to the topology character of *k*-ary *n*-trees, we can divide all switches at stage *l* into k^l equivalence-classes base on relation *R*. There are k^{n-1-l} switches in each equivalence class. The equivalence class to which the switch *SW* belongs is recorded as [*SW*]. For example, all switches in stage 1 of the 4-ary 3-tree

(as shown in Figure 1) can be divided into 4 equivalence classes: {<0,0,1>, <1,0,1>, <2,0,1>, <3,0,1>}, {<0,1,1>, <1,1,1>, <2,1,1>, <3,1,1>}, {<0,2,1>, <1,2,1>, <2,2,1>, <3,2,1>} and {<0,3,1>, <1,3,1>, <2,3,1>, <3,3,1>}.

Theorem 1: Given an k-ary n-tree $FT(k, n)$, P is a path from the source processing node S to the destination processing node D. Suppose P passes through the switch $SW=<\omega_0,\omega_1,\ldots, \omega_{n-2}, l>$ and switch $SW'=<\omega_0',\omega_1',\ldots, \omega_{n-2}', l>$ in the ascending and descending phase respectively, then $SW\ R\ SW'$.

Proof: Let the switch $SW_{lca}=<s_0,s_1,\ldots, s_{n-2}, m>$ be the lowest common ancestor of S and D through which the path P passes. The path P_s from S to SW_{lca} passes through $SW_S=<\omega_0,\omega_1,\ldots, \omega_{n-2}, m+1>$, and the path P_D from SW_{lca} to D passes through $SW_D=<\omega_0',\omega_1',\ldots, \omega_{n-2}', m+1>$, according to the definition of k-ary n-tree we can easily get that $\omega_i=\omega_i'$ for all $i\neq m$, then $|gcs(SW_S,SW_D)|=n-1-(m+1)$. The rest may be deduced by analogy, if P passes through the switch $SW=<\omega_0,\omega_1,\ldots, \omega_{n-2}, l>$ and $SW'=<\omega_0',\omega_1',\ldots, \omega_{n-2}', l>$, then $\omega_i=\omega_i'$ for all $i\notin[m,l-1]$, therefore, $|gcs(SW,SW')| = n-1-l$ and $SW\ R\ SW'$.

Following from Theorem 1, we can easily get Lemma 1.

Lemma 1: Given a k-ary n-tree $FT(k, n)$, suppose a multicast tree passes through the switch $SW=<\omega_0,\omega_1,\ldots, \omega_{n-2}, l>$ at the stage l , then $SW\ R\ SW'$ for each $SW'=<\omega_0',\omega_1',\ldots, \omega_{n-2}', l>$ through which the multicast tree passes.

Theorem 1 and lemma 1 show that the switches, which are at the same stage and are passed through by a unicast routing path or a multicast tree, belong to the same equivalence class.

3.2 Conflict-Free Routing

Theorem 1 and Lemma1 indicates that if packet p_1 conflicts with packet p_2 in the descending phase at stage l, then p_1 and p_2 must passes through the switches belonging to the same equivalence class at stage l in the ascending phase. From the switch labeling strategy and Definition 2 we can see that if two packets p_1 and p_2 don't pass through switches belonging to the same equivalence class at stage l, then these two packets don't pass through switches belonging to the same equivalence at stage l' for all $l'\leq l$. This characteristic of k-ary n-tree is shown in Lemma 2.

Lemma 2: Given an k-ary n-tree $FT(k, n)$, suppose two packets (unicast or multicast), p_1 and p_2, pass through $SW=<\omega_0,\omega_1,\cdots, \omega_{n-2}, l>$ and $SW'=<\omega_0',\omega_1',\cdots, \omega_{n-2}', l>$ in the ascending phase respectively, if $[SW]\neq[SW']$, then no conflict between m_1 and m_2 will occurs at stage l' for all $l'\leq l$. If $l=n-2$ and the destination (destinations) of p_1 is not overlap with that (those) of p_2, then no conflict will occurs between p_1 and p_2 in the whole network.

3.3 Look-Ahead

For a multicast tree, the workload of a branch is close to that of others. Therefore, if some link in a certain multicast tree is heavy loaded, to some degree, we can say that the multicast tree is heavy loaded. Following from lemma 2, to reduce the conflict between unicast and multicast packets, an ascending unicast packet would better

doesn't use the uplink which leads to a switch that is passed through by a heavy loaded multicast tree, which is the basic idea of look-ahead adaptive routing.

Here we suppose that the wormhole switching technology is employed and a packet is divided into a number of fixed size flits to be transmitted one after another. The *multicast-load* of a link can be defined as the number of multicast flits being hold in the input buffer associating with the link. Instead of allowing ascending unicast packet to use any idle output port, look-ahead adaptive routing strategy forbids ascending unicast packet using the output ports associating with heavy loaded link. A question that arises is under which circumstance a downlink can be considered heavy loaded. What we need is a threshold value: if the link load exceeds this threshold, then the link is considered to be heavily loaded. The selecting of the threshold value is an important issue. If the threshold value is too small, the number of links that are forbidden being used by ascending unicast packet is large, which may lead to higher unicast latency. Otherwise, a large threshold value leads to higher multicast latency. If the threshold value is larger than the input buffer length, look-ahead adaptive routing is equal to full adaptive routing.

Let us give an example to explain the look-ahead adaptive routing strategy. Given a 4-ary 3-tree and a process groups G_1={<1,0,0>,<1,2,2>,<2,0,2>,<2,3,0>}, the multicast tree used by G_1 (recorded as T_1) is shown in Fig 2. Suppose a unicast packet from processing node <1,0,3> to processing node <2,3,2> is requesting for the up-ports of switch <1,0,2> and the link e_1 is heavy loaded. If the packet is sent to up-port associated with e_1, based on theorem 1, the packet will pass through switch <2,0,1> in the descending phase, and conflict between the unicast packet and the multicast packet being transmitted in T_1 may occurs at edge e_2. Otherwise, this conflict on e_2 doesn't occur.

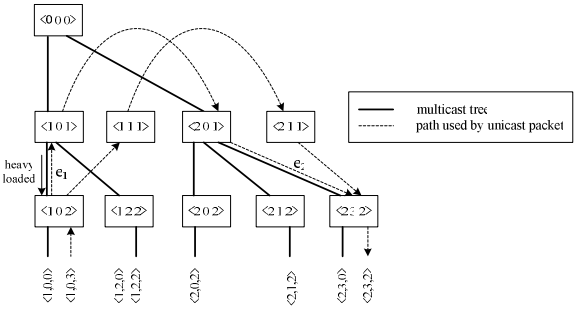


Fig. 2. An example of look-ahead adaptive routing

4 Evaluation

To investigate the performance of the proposed look-ahead adaptive routing algorithm, a discrete event-driven simulator has been built using the *OMNeT++* [12] simulation platform. The simulation is observed by the tool *Akaroa* [13]. The simulation model and results are described in this section.

4.1 Simulation Model

To accurately measure the communication latency, we employ a flit level simulation. The time taken by a flit to be forwarded from one switch to another is assumed to be one simulation time unit (we call it as one *cycle*). To improve the traffic flow, we have considered four unicast virtual channels and one multicast virtual channel per physical channel with 32 flits buffer per virtual channel. A fixed packet size of 8 flits is assumed. A multicast packet must wait in the input buffer until being delivered to all destination ports. To achieve higher switch throughput, the fan-out splitting of multicast packet is allowed. The integrated arbitration strategy used in the simulator is MURS-mix [14], but here we use WBA [15] strategy for multicast arbitration instead of RR.

The packet arrival time at each processing node is assumed to be uniformly distributed. The destination of unicast and multicast packets are also assumed to be uniformly distributed. To compare the performance in different group sizes, we have used the normalized network load to determine the generation rate of the packet. The normalized network load, denoted by *Traffic*, is the amount of flits delivered per time unit per processing node. The multicast delay refers to the interval between the packet initiation time and the time until all destinations receive the entire packet. If there is more than one flit in a packet, the head flit is followed by several body flits and a tail flit. Otherwise, the head flit is also a tail flit. The injection rate of unicast head flit (r_u) and multicast head flit (r_m) can be obtained using the formula 1 and formula 2 respectively, where M_u (M_m) denotes the ratio unicast (multicast) to network load, L_u (L_m) denotes the length of unicast (multicast) packet, S_g denotes the size of the multicast group.

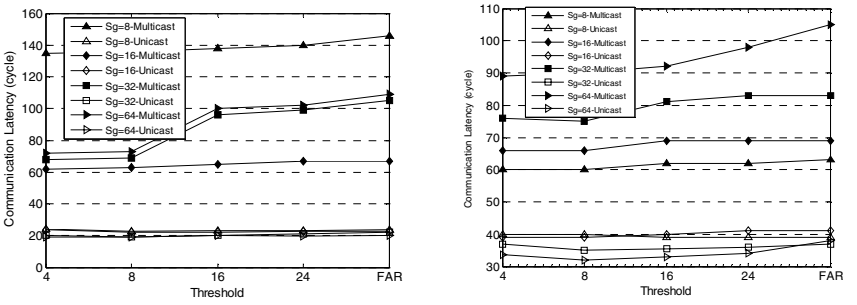
$$r_u = \frac{traffic \cdot M_u}{(1 - traffic \cdot M_u) \cdot L_u + traffic \cdot M_u} \tag{1}$$

$$r_m = \frac{traffic \cdot M_m}{((1 - traffic \cdot M_m) \cdot L_m + traffic \cdot M_m) \cdot S_g} \tag{2}$$

4.2 Simulation Results

In our experiment, a 4-ary 3-tree is simulated, and the performance of look-ahead adaptive routing (LAR) is compared with that of full adaptive routing (FAR). The size of multicast group is set to 8, 16, 32, and 64. The M_m is set to 0.5 and 0.2 separately. As simulator termination criteria, the max relative error is set to 5%, and the confidence level is set to 95%.

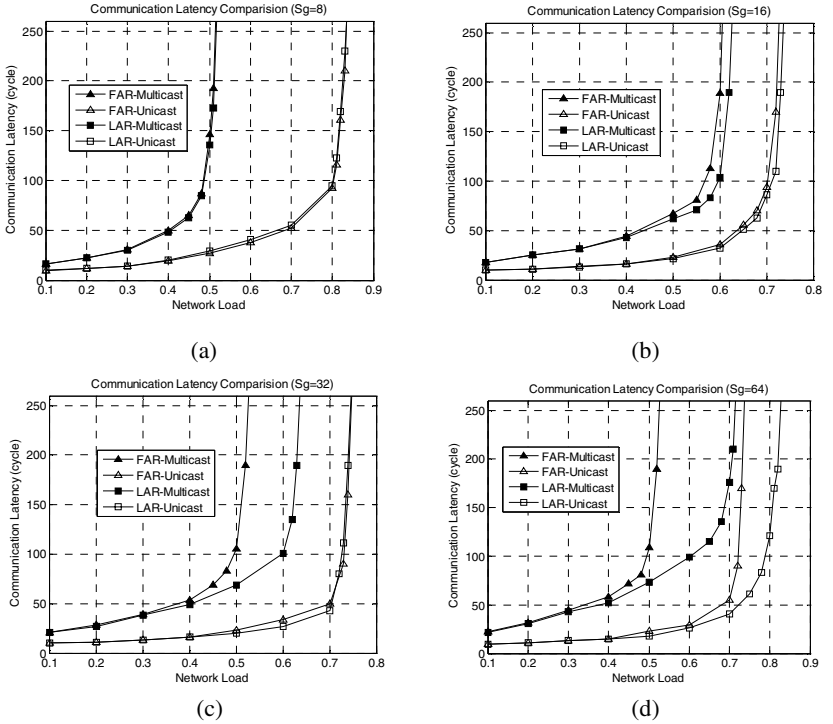
Firstly, we perform experiments to determine the optimum value for the threshold. Fig 3 shows the multicast and unicat latency under different threshold value when the M_m =0.5 (*Traffic*=0.5) and M_m =0.2 (*Traffic*=0.6). If threshold is larger than 32, LAR is equal to FAR. From the latency curve we see that, for good performance, the threshold should be 8. This value will be used throughout the rest of the paper.



(a) (b)

Fig. 3. The impact of link threshold on communication latency: (a) *Traffic*=0.5, M_m =0.6, (b) *Traffic*=0.6, M_m =0.2

In the first set of result, the M_m is set to 0.5. Fig 4 shows the communication latency versus the network load. When S_g =8, the performance of LAR is very close to that of FAR. When S_g =16, the multicast latency of LAR is a little lower than that of FAR. For large group size (32 and 64), the multicast throughput of LAR is higher than that of FAR by about 20%. Fig.4 indicates that for higher M_m, LAR can also



(a) (b)

(c) (d)

Fig. 4. Communication latency comparison of FAR and LAR (M_m =0.5)

lower the unicast latency compared with FAR. Especially, the unicast throughput of LAR is higher than that of FAR by about 15% when S_g =64.

In most classes of parallel applications, multicast communication constitutes only a small portion of the total network traffic. In the next set of simulation results, the M_m is set to 0.2. Fig. 5 shows that the multicast throughput of LAR is higher than that of FAR by about 10%, 25%, 27%, and 30% respectively for S_g =8, 16, 32, and 64. The unicast latency of LAR is also a little lower than that of FAR when S_g is larger than 16.

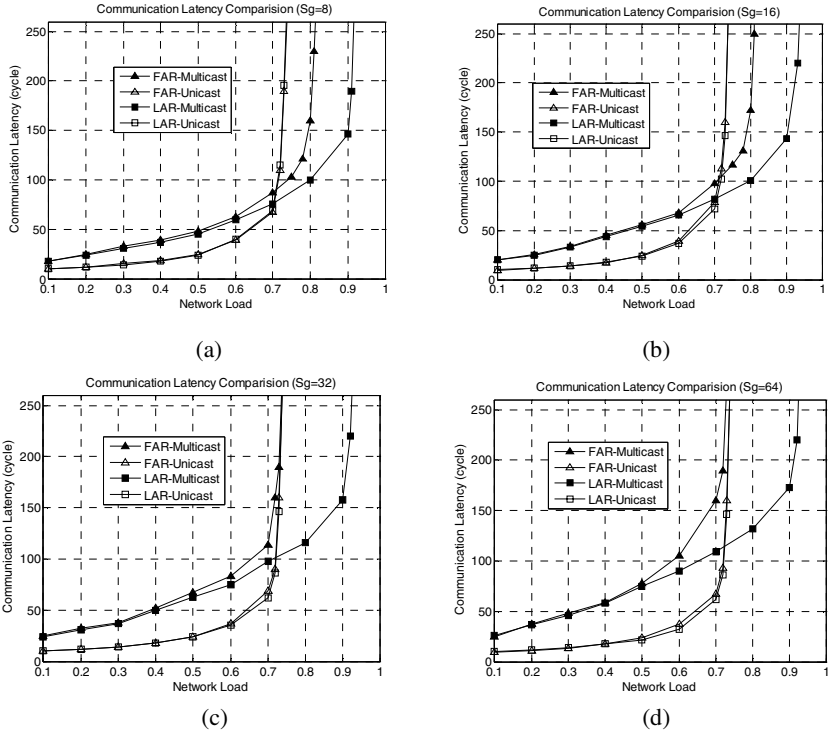


Fig. 5. Communication latency comparison of FAR and LAR (M_m =0.2)

5 Conclusion

In this paper we study the methods to forecast and reduce the conflict between unicast and multicast on *k*-ary *n*-trees. Based on the switch grouping method, we analysis the sufficient conditions for no conflict routing and propose the look-ahead adaptive routing strategy. The simulation results indicate that the proposed strategy can lower the multicast latency and the unicast latency simultaneously.

References

1. Petrini, F., Vanneschi, M.: k-ary n-trees: High Performance Networks for Massively Parallel Architecture. In: International Parallel Processing Symposium, Geneva, Switzerland, pp. 87–93 (April 1997)
2. Petrini, F., Feng, W.c., Hoisie, A., Coll, S., Frachtenberg, E.: The Quadrics Network: High Performance Clustering Technology. IEEE Micro 22(1), 46–57 (2002)
3. Panda, D.K.: Issues in Designing Efficient and Practical Algorithms for Collective Communication in Wormhole Routed Systems. In: 1995 WorkShop on Challenges for Parallel Processing, pp. 8–15 (1995)
4. Ni, L.M.: Should Scalable Parallel Computer support Efficient Hardware Multicast? In: Proceeding of the ICPP Workshop on Challenges for Parallel Processing, pp. 2–7 (1995)
5. http://www.supercomp.de/isc2005/index.php?s=tutorial&unterseite=overview
6. Panda, D.K., Sivaram, R.: Fast Broadcast and Multicast in Wormhole Mutistage Networks with Multidestination worms. Tech. Rep. OSU-CISRC-04/95-TR21, Department of Computer and Information Science, Ohio State University (1995)
7. Varavithya, V., Mohapatra, P.: Tree-Based Multicasting on Wormhole Routed Multistage Interconnection Networks. In: Proceedings of the International Conference on Parallel Processing (1997)
8. Kumar, S.: Optimizing Communication for Massively Parallel Processing, PhD thesis, department of computer science, university of Illinois at Urbana-Champaign (May 2005)
9. Sun, Q., Zhang, M., Xiao, L.: Hardware-Based Multicast with Global Load Balance on k-ary n-trees, in Proceedings of the International Conference on Parallel Processing (2007)
10. Aydogan, Y.: Adaptive Source Routing and Route Generation for Multicomputers, Master thesis, department of computer engineering and information science. Bilkent University
11. Lin, X., Ni, L.M.: Deadlock-free Multicast Wormhole Routing in Multicomputer Networks. In: Proceedings of the International Symposium on Computer Architecture, pp. 116–124 (1991)
12. http://www.omnetpp.org/
13. http://www.cosc.canterbury.ac.nz/research/RG/net_sim/simulation_group/akaroa/
14. Mhamdi, L., Vassiliadis, S.: Integrating Uni- and Multicast Scheduling in Buffered Crossbar Switches. In: Workshop on High Performance Switching and Routing (2006)
15. Prabhakar, B., Mckeown, N., Ahuja, R.: Multicast Scheduling for Input-Queued Switches. IEEE Journal of Selected Arears Communication 15(5), 855–866 (1997)

A Beehive Algorithm Based QoS Unicast Routing Scheme with ABC Supported*

Xingwei Wang, Guang Liang, and Min Huang

College of Information Science and Engineering, Northeastern University, Shenyang, 110004, P.R. China
wangxw@mail.neu.edu.cn

Abstract. In this paper, a QoS unicast routing scheme with ABC supported is proposed based on beehive algorithm. It deals with inaccurate network status information and imprecise user QoS requirement, introduces edge bandwidth pricing, edge evaluation and path evaluation, and tries to find a QoS unicast path with Pareto optimum under Nash equilibrium on both the network provider utility and the user utility achieved or approached.

1 Introduction

QoS (Quality of Service) routing with ABC (Always Best Connected) supported is essential [1]. However, it is hard to describe the network status exactly. The user QoS requirements are affected largely by a lot of subjective factors and often can not be expressed accurately. ABC means a user can get the best available connection anytime and anywhere, however, 'best' itself is fuzzy, depending on many factors, such as user QoS requirement, cost a user willing to pay, user preference, terminal ability and access network availability. With network operation commercialization, ABC is not a user's own wishful thinking and thus need to consider both the network provider profit and the user profit with win-win supported [2]. In this paper, a QoS unicast routing scheme with ABC supported is proposed based on the beehive algorithm [3], trying to find a QoS unicast path with Pareto optimum under Nash equilibrium on both the network provider utility and the user utility achieved or approached.

2 Model Description

A network is represented as a graph $G(V,E)$, V is node set and E is edge set. $\forall v_i, v_j \in V(i, j = 1,2,3,\ldots,|V|)$, there maybe exist several edges between them. The node parameters are merged into the edge ones. $\forall e_l \in E$, consider total bandwidth

* This work is supported by the National High-Tech Research and Development Plan of China under Grant No. 2006AA01Z214; the National Natural Science Foundation of China under Grant No. 60673159; Program for New Century Excellent Talents in University; Specialized Research Fund for the Doctoral Program of Higher Education; the Natural Science Foundation of Liaoning Province under Grant No. 20062022.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 450–459, 2007.

bwt_l, available bandwidth bw_l, delay dl_l, delay jitter jt_l, error rate ls_l, bandwidth unit cost ct_l and bandwidth price p_l. A QoS unicast routing request is described as $<v_s,v_d,[bw\_rq_L,bw\_rq_H],[dl\_rq_L,dl\_rq_H],[jt\_rq_L,jt\_rq_H],[ls\_rq_L,ls\_rq_H],py,bd>$, its elements representing the source node, the destination node, the user bandwidth, delay, delay jitter and error rate requirement, the cost the user willing to pay and the bid the user willing to offer respectively. $py\in\{py_e,py_g,py_f,py_p\}$ and $bd\in\{bd_e,bd_g,bd_f,bd_p\}$ correspond to the specific cost the user willing to pay and the specific bid the user willing to offer respectively when the user QoS level takes a specific value from {Excellent,Good,Fair,Poor}.

In order to promote a user to consume bandwidth rationally, the edge bandwidth price can be divided into three different regions, i.e., low, sound, and high. Assume η_l represents the loading level of e_l and is computed as follows:

$$\eta_l = \frac{bwt_l - bw_l}{bwt_l} \quad . \tag{1}$$

If $\eta_l < \eta_0$, e_l is considered to be low-loaded, its corresponding bandwidth price is at the low region and can be decreased according to the formula (2); if $\eta_l > \eta_1$, e_l is considered to be high-loaded, its corresponding bandwidth price is at the high region and can be increased according to the formula (3); otherwise, e_l is considered to be moderate-loaded and its corresponding bandwidth price is at the sound region. Here, η_0 and η_1 are preset experience value, $0<\eta_0<\eta_1<1$.

$$p_l = \begin{cases} p_l^{\min} & \eta_l < \eta_{\min} \\ \dfrac{A}{1+\alpha\times\eta_l^{-\beta}} & \eta_l^{\min} \le \eta_l \le \eta_0 \end{cases} \quad . \tag{2}$$

$$p_l = \begin{cases} p_l^{\max} & \eta_{\max} < \eta_l < \eta_1 \\ B\times\left(2-e^{-\delta\times(\eta_l-\eta_0)^2}\right) & \eta_0 \le \eta_l \le \eta_{\max} \end{cases} \quad . \tag{3}$$

$p_l^{\min}$ and $p_l^{\max}$ are the lower bound of the low region and the upper bound of the high region respectively, $\eta_{\min}$ represents the upper bound of the edge loading level corresponding to $p_l^{\min}$, and $\eta_l^{\max}$ represents the lower bound of the edge loading level corresponding to $p_l^{\max}$.

Assume p_l^0 represents the bandwidth baseline price of e_l, η_l^0 is the edge loading level corresponding to p_l^0, $p_l^{\min}\le p_l^0\le p_l^{\max}$.

For $\eta_{\min}\le\eta_l\le\eta_0$, p_l is Cauchy distribution alike[4] with β=2, $p_l=p_l^0$ when $\eta_l=\eta_0$, $p_l=p_l^{\min}$ when $\eta_l=\eta_{\min}$. Then, A and α can be derived as follows:

$$p_l^0 = \frac{A}{1+\alpha\times\eta_0^{-2}} \quad . \tag{4}$$

$$p_l^{\min} = \frac{A}{1+\alpha \times \eta_{\min}^{-2}} \ . \tag{5}$$

$$A = p_l^0 \times (1 + \frac{p_l^0 - p_l^{\min}}{p_l^{\min} \times \eta_{\min}^{-2} - p_l^0 \times \eta_0^{-2}} \times \eta_0^{-2}) \ . \tag{6}$$

$$\alpha = \frac{p_l^0 - p_l^{\min}}{p_l^{\min} \times \eta_{\min}^{-2} - p_l^0 \times \eta_0^{-2}} \ . \tag{7}$$

For $\eta_0 \le \eta_l \le \eta_{\max}$, p_l is normal distribution alike[4], $p_l = p_l^0$ when $\eta_l = \eta_0$, $p_l = p_l^{\max}$ when $\eta_l = \eta_{\max}$. Then, B and δ can be derived as follows:

$$p_l^0 = B \times \left(2 - e^{-\delta \times (\eta_0 - \eta_0)^2}\right) . \tag{8}$$

$$p_l^{\max} = B \times \left(2 - e^{-\delta \times (\eta_{\max} - \eta_0)^2}\right) . \tag{9}$$

$$B = p_l^0 \ . \tag{10}$$

$$\delta = -\frac{\ln(2 - \frac{p_l^{\max}}{p_l^0})}{(\eta_{\max} - \eta_0)^2} \ . \tag{11}$$

The membership degree function is introduced to describe the adaptability of the candidate edge conditions to the user QoS requirements. The edge bandwidth, delay, delay jitter and error rate adaptability membership degree function are defined as the following formulas (12) to (15) respectively.

$$g_1(abw_l) = \begin{cases} 2 \times (\frac{abw_l - bw\_rq_L}{bw\_rq_H - bw\_rq_L})^2 & bw\_rq_L < abw_l \le \frac{1}{2}(bw\_rq_H + bw\_rq_L) \\ 1 - 2 \times (\frac{bw\_rq_H - abw_l}{bw\_rq_H - bw\_rq_L})^2 & \frac{1}{2}(bw\_rq_H + bw\_rq_L) < abw_l \le bw\_rq_H \\ 1 & abw_l > bw\_rq_H \end{cases} . \tag{12}$$

$$g_2(dl_l) = \begin{cases} 1 & dl_l \le dl\_rq_L \\ (\frac{dl\_rq_H - dl_l}{dl\_rq_H - dl\_rq_L})^k & dl\_rq_L < dl_l \le dl\_rq_H \\ 0 & dl_l > dl\_rq_H \end{cases} . \tag{13}$$

$$g_3(jt_l) = \begin{cases} 1 & jt_l \le jt\_rq_L \\ \left(\dfrac{jt\_rq_H - jt_l}{jt\_rq_H - jt\_rq_L}\right)^k & jt\_rq_L < jt_l \le jt\_rq_H \\ 0 & jt_l > jt\_rq_H \end{cases} . \tag{14}$$

$$g_4(ls_l) = \begin{cases} 1 & ls_l \le ls\_rq_L \\ \left(\dfrac{ls\_rq_H - ls_l}{ls\_rq_H - ls\_rq_L}\right)^k & ls\_rq_L < ls_l \le ls\_rq_H \\ 0 & ls_l > ls\_rq_H \end{cases} . \tag{15}$$

The evaluation matrix $G=[g_1 \quad g_2 \quad g_3 \quad g_4]^T$ to e_l can be gotten by the formulas (12) to (15). According to the application nature, the corresponding weight matrix $\Lambda=[\lambda_1 \quad \lambda_2 \quad \lambda_3 \quad \lambda_4]$ is given, λ_1 , λ_2 , λ_3 and λ_4 are the relative significance weights of bandwidth, delay, delay jitter and error rate to the application QoS respectively, $0 \le \lambda_1, \lambda_2, \lambda_3, \lambda_4 \le 1$, $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$. Then, the user satisfaction degree to the QoS of e_l is computed as follows:

$$St_l = \Lambda \circ G \ . \tag{16}$$

The mapping relationship between St_l and Ql is defined as follows:

$$Ql = \begin{cases} Excellent & St_l \ge \alpha_1 \\ Good & \alpha_2 \le St_l < \alpha_1 \\ Fair & \alpha_3 \le St_l < \alpha_2 \\ Poor & St_l < \alpha_3 \end{cases} . \tag{17}$$

α_1 , α_2 and α_3 are preset experience values. According to Ql , the corresponding py value can be determined from $\{py_e, py_g, py_f, py_p\}$.

Define the user satisfaction degree to the cost he paid as follows:

$$Dp_l = \begin{cases} 1 & p_l \times abw_l \le \dfrac{1}{\chi} py \\ \left(\dfrac{py - p_l \times abw_l}{py - \dfrac{1}{\chi} py}\right)^k & \dfrac{1}{\chi} py < p_l \times abw_l \le py \\ 0 & p_l \times abw_l > py \end{cases} . \tag{18}$$

abw_l is the actually allocated bandwidth for the user on e_l, $bw\_rq_L \le abw_l \le bw\_rq_H$; $\chi > 1$, k is a preset experience value.

The user utility on e_l is computed as follows:

$$uu_l = \varpi_1 \times St_l + \varpi_2 \times Dp_l \ . \tag{19}$$

ϖ_1 and ϖ_2 are preference weights for St_l and Dp_l respectively, $0 \le \varpi_1, \varpi_2 \le 1$, $\varpi_1 + \varpi_2 = 1$.

$\forall e_l \in E$, the initial value of its being selected probability is computed as follows:

$$pr_l(0) = \frac{1}{se} \ . \tag{20}$$

Here, se is the number of those edges sharing the same endpoints with e_l, that is, all edges sharing the same endpoints have the same being selected probability at the beginning in this paper. Since the proposed scheme in this paper is based on the bee-hive algorithm [3], when routing, the edge being selected probability is increased or decreased through feedback by means of judging the reaction valve values of the bees themselves and the external stimulation signal values. After the $(bn+1)$ th bee is produced, the being selected probability $pr_l(bn+1)$ of e_l is updated as follows:

$$pr_l(bn+1) = \frac{S_{bn+1}^2}{S_{bn+1}^2 + \theta_{bn+1}^2} \ . \tag{21}$$

$$S_{bn+1} = \begin{cases} b_1 S_0 + b_2 * \dfrac{\tau}{Hop_{bn+1}(he(e_l), v_t)} & \text{the } (bn+1)th \text{ bee passing } e_l \\ S_0 & otherwise \end{cases} \ . \tag{22}$$

$$\theta_{bn+1} = \begin{cases} f_1 \theta_0 + f_2 * \lambda * Hop_{bn+1}(he(e_l), v_t) & \text{the } (bn+1)th \text{ bee passing } e_l \\ \theta_0 + \sigma * Hop_{bn+1}(he(e_l), v_t) & \text{otherwise} \end{cases} \ . \tag{23}$$

Here, S_{bn+1} is the stimulation signal value produced by $(bn+1)$ th bee, S_0 is its baseline value, τ is a constant, $\tau > 1$, b_1 and b_2 are its preference weights, $0 \le b_1, b_2 \le 1$, $b_1 + b_2 = 1$, $he(e_l)$ is the starting end node of e_l, $Hop_{bn+1}(he(e_l), v_t)$ represents the distance (hop number) between $he(e_l)$ and v_t along the path from v_s to v_t traversed by the $(bn+1)$ th bee; θ_{bn+1} is the reaction valve value to v_d of the $(bn+2)$ th bee itself, θ_0 is its baseline value, λ and σ are constants, $0 < \lambda < 1$, $0 < \sigma < 1$, f_1 and f_2 are its preference weights, $0 \le f_1, f_2 \le 1$, $f_1 + f_2 = 1$.

Assume that there are k candidate edges between two nodes provided by different network providers and consider the following seven attributes for each edge: the available bandwidth, delay, delay jitter, error rate, loading level, bandwidth price and being selected probability, constitute a $k \times 7$ evaluation matrix as follows:

$$F = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{17} \\ f_{21} & f_{22} & \cdots & f_{27} \\ \cdots & \cdots & \cdots & \cdots \\ f_{k1} & f_{k2} & \cdots & f_{k7} \end{bmatrix} . \tag{24}$$

f_{zy} represents the yth attribute of the zth candidate edge, $1 \le z \le k, 1 \le y \le 7$. $f_{\min,y}$ and $f_{\max,y}$ are the minimum and maximum of the yth attribute value respectively. In order to eliminate the influence of the attribute magnitude and reserve the information about the attribute variation value, do normalization to "the larger the better" and "the smaller the better" attributes according to the formulas (25) and (26) respectively:

$$f_{zy}^{'} = \frac{f_{zy}}{f_{\max,y} + f_{\min,y}} . \tag{25}$$

$$f_{zy}^{'} = \frac{f_{\max,y} + f_{\min,y} - f_{zy}}{f_{\max,y} + f_{\min,y}} . \tag{26}$$

$$f_{\min,y} = \min_{z}\{f_{zy}\} . \tag{27}$$

$$f_{\max,y} = \max_{z}\{f_{zy}\} . \tag{28}$$

Based on the above, the network provider utility is computed as follows:

Step1: Get the network providers of k candidate edges and construct the corresponding $k \times 7$ evaluation matrix F.

Step2: Modify F: normalize the delay, delay jitter, error rate and loading level attribute according to the formula (26); normalize the available bandwidth, bandwidth price and being selected probability attribute according to the formula (25).

Step3: Compute the standard deviation s_y for each attribute according to the formula (29); compute the weight for each attribute according to the formula (31).

$$s_y = \left(\frac{1}{k} \times \sum_{z=1}^{k} (f_{zy}^{'} - \overline{f_y})^2 \right)^{\frac{1}{2}} . \tag{29}$$

$$\overline{f_y} = \frac{1}{k} \times \sum_{z=1}^{k} f_{zy}^{'} . \tag{30}$$

$$w_y = \frac{s_y}{\sum_{\kappa=1}^{t} s_\kappa} . \tag{31}$$

Step4: Compute the zth network provider utility nu_l^z on its provided candidate edge e_l according to the formula (32).

$$nu_l^z = \sum_{y=1}^{t} w_y \times f_{zy}^{'} \quad . \tag{32}$$

The network provider and the user play game on an edge. The network provider has two gaming strategies: whether it is willing to provide the bandwidth of the candidate edge to the user or not. The user also has two gaming strategies: whether he is willing to accept the provided bandwidth of the candidate edge or not. The gaming matrixes of the user and the zth network provider on its provided candidate edge e_l are defined as follows respectively:

$$UU = \begin{bmatrix} \ln \dfrac{uu_l}{uu_{l_0}} & \gamma \times \ln \dfrac{uu_l}{uu_{l_0}} \\ -\mu \times \ln \dfrac{uu_l}{uu_{l_0}} & 0 \end{bmatrix} . \tag{33}$$

$$NU^z = \begin{bmatrix} \ln \dfrac{nu_l^z}{nu_{l_0}^z} & -\mu \times \ln \dfrac{nu_l^z}{nu_{l_0}^z} \\ \gamma \times \ln \dfrac{nu_l^z}{nu_{l_0}^z} & 0 \end{bmatrix} . \tag{34}$$

The rows in NU^z and UU correspond to the user gaming strategies: accept or not, the columns correspond to the zth network provider gaming strategies: provide or not. $nu_{l_0}^z$ and uu_{l_0} represent the lowest acceptable utilities of the zth candidate network provider and the user on the candidate edge e_l respectively, which are preset experience values. nu_l^z and uu_l represent the zth network provider utility and the user utility on e_l. If the zth network provider is willing to provide e_l but the user rejects it or the user is willing to accept e_l but the zth network provider does not provide it, the user or the zth network provider will be punished, μ is a penalty factor bigger than 1. If the zth network provider does not provide e_l but the user is willing to accept it or the user rejects e_l but the zth network provider is willing to provide it, the user or the zth network provider will lose its utility, γ is a loss factor smaller than 1. If the zth network provider does not provide e_l at the same time the user rejects it, their utilities are 0. If the strategy pair $<p^*,q^*>$ satisfies the following inequality, the strategy pair $<p^*,q^*>$ is the solution under Nash equilibrium [5]:

$$\begin{cases} a_{p^*q^*} \geq a_{pq^*} \\ b_{p^*q^*} \geq b_{p^*q} \end{cases} . \tag{35}$$

If $<p^*,q^*>$ is <provide,accept>, e_l will be selected, otherwise rejected.

The user utility UU_{sd}, the zth network provider utility NU_{sd}^{z} and all network provider utility NU_{sd} on the path P_{sd} from v_s to v_d are computed as follows:

$$UU_{sd} = \sum_{e_l \in P_{sd}} uu_l \ . \tag{36}$$

$$NU_{sd}^{z} = \sum_{e_l \in P_{sd}} nu_l^{z} \ . \tag{37}$$

$$NU_{sd} = \frac{1}{\sum_{z} \frac{1}{NU_{sd}^{z}}} \ . \tag{38}$$

The cost of the path P_{sd} is computed as follows:

$$CT_{sd} = \sum_{e_l \in P_{sd}} ct_l \times abw_l \ . \tag{39}$$

Considering the user utility, the network provider utility and the path cost comprehensively, the path evaluation function $J_{P_{sd}}$ of P_{sd} is defined as follows:

$$J_{P_{sd}} = \beta_1 \times \frac{\Omega_1}{UU_{sd}} + \beta_2 \times \frac{\Omega_2}{NU_{sd}} + \beta_3 \times \frac{CT_{sd}}{\Omega_3} \ . \tag{40}$$

Here, β_1, β_2 and β_3 are the preference weights to the user utility, the network provider utility and the path cost respectively, $0 \le \beta_1, \beta_2, \beta_3 \le 1$, $\beta_1 + \beta_2 + \beta_3 = 1$; Ω_1, Ω_2 and Ω_3 are tuning coefficients, making Ω_1 / UU_{sd}, Ω_2 / NU_{sd} and CT_{sd} / Ω_3 into the same magnitude order. According to the formula (40), the smaller the value of $J_{P_{sd}}$, then the bigger the value of UU_{sd}, NU_{sd}^{z}, NU_{sd} and $(UU_{sd} + NU_{sd})$, the much possible for them to achieve or approach Pareto optimum under Nash equilibrium, and achieve the smaller value of CT_{sd}.

3 Algorithm Description

In this paper, the objective of the proposed scheme is described as follows:

$$\text{maxmize}\{UU_{sd}\} \ . \tag{41}$$

$$\text{maxmize}\{NU_{sd}^{z}\} \ . \tag{42}$$

$$\text{maxmize}\{NU_{sd}\} \ . \tag{43}$$

$$\text{maxmize}\{UU_{sd} + NU_{sd}\} \ . \tag{44}$$

$$\text{minmize}\{CT_{sd}\} \ . \tag{45}$$

s.t.

$$\min_{e_l \in P_{sd}}\{abw_l\} \geq bw\_rq_L \ . \tag{46}$$

$$\sum_{e_l \in P_{sd}} dl_l \leq dl\_rq_H \ . \tag{47}$$

$$\sum_{e_l \in P_{sd}} jt_l \leq jt\_rq_H \ . \tag{48}$$

$$1 - \prod_{e_l \in P_{sd}} (1 - ls_l) \leq ls\_rq_H \ . \tag{49}$$

The proposed QoS unicast routing algorithm based on the beehive algorithm in this paper is described as follows:

Step1: Initialization: set the maximum number of the bees BN (covering both the short distance bees and the long distance bees [3]); the number of the produced bees by far $bn = 0$; the baseline bid bd_l^0 of the user to e_l; computing the initial value of the edgee $e_l's$ being selected probability; the period of the long distance bees being produced itv; the life cycle of the long distance bees ltv; the life cycle of the short distance bees stv; the counter of the hop number $hp = 0$; the current best path $Pt_{best}=\varphi$ and its evaluation value $J_{P_0}=\infty$.

Step2: If $bn < BN$, go to Step3; otherwise, go to Step18.

Step3: If hp can be exactly divided by itv, produce the long distance bee Be_{bn} from v_s; otherwise, produce the short distance bee Be_{bn}.

Step4: Set the followings carried by Be_{bn}: $bw=\infty$, $dl=0$, $jt=0$, $ls=0$, the current node $cn=v_s$, path $Pt=\{v_s\}$, the set of those edges connecting with cn: $ne=\varphi$, the set of the candidate edges for the next hop: $ce=\varphi$.

Step5: If $cn=v_i$, put all edges connecting with v_i into ne and delete those edges from ne which cause loop, that is, $ne=ne-\{e_l \,|\, e_l \in ne \wedge ta(e_l) \in Pt\}$, $ta(e_l)$ is another endpoint of e_l besides v_i.

Step6: If Be_{bn} is a long distance bee, go to Step7, otherwise go to Step8.

Step7: If $hp > ltv$, Be_{bn} died, go to Step17; otherwise, go to Step9.

Step8: If $hp > stv$, Be_{bn} died, go to Step17; otherwise, go to Step9.

Step9: If $ne=\varphi$, go to Step14, otherwise select one e_l from ne at random.

Step10: If $\min\{bw,abw_l\}<bw\_rq_L$ or $dl+dl_l>dl\_rq_H$ or $jt+jt_l>jt\_rq_H$ or $1-(1-ls)(1-ls_l)>ls\_rq_H$, $ne=ne-\{e_l\}$, go to Step9; otherwise, compute the loading level of e_l.

Step11: If e_l is high-loaded, compute the user satisfaction degree St_l to the QoS of e_l, get its corresponding QoS level Ql, determining the user`s specific bd from $\{bd_e, bd_g, bd_f, bd_p\}$ to e_l, compute p_l according to the formula (3), go to Step12;

if e_l is low-loaded, compute p_l according to the formula (2); if e_l is moderate-loaded, go to Step13.

Step12: If $bd<bd_l^0$, $ne=ne-\{e_l\}$, go to Step9; otherwise, $p_l=bd$, $bd_l^0=bd$.

Step13: Compute the user edge utility uu_l and the network provider edge utility nu_l^z; the user and the network provider play game on the edge e_l: if Nash equilibrium is achieved and its corresponding gaming strategy is <provide,accept>, $ce=ce\cup\{e_l\}$, $ne=ne-\{e_l\}$, otherwise $ne=ne-\{e_l\}$; go to Step9.

Step14: If $ce=\varphi$, Be_{bn} died, go to Step17; otherwise, according to $pr_l(bn)$, select one e_l from ce as the next hop, do: $bw=\min\{bw,abw_l\}$, $dl=dl+dl_l$, $jt=jt+jt_l$, $ls=1-(1-ls)(1-ls_l)$, $cn=ta(e_l)$, $Pt=Pt\cup\{e_l\}\cup\{cn\}$, $ne=\varphi$, $ce=\varphi$, $hp=hp+1$.

Step15: If $cn\neq v_d$, go to Step5.

Step16: Compute the user utility UU_{sd}, the zth network provider utility NU_{sd}^z and the network provider utility NU_{sd} respectively; compute the path cost CT_{sd}; compute the path evaluation value $J_{P_{sd}}$. If $J_{P_{sd}}<J_{P_0}$, $Pt_{best}=Pt$, $J_{P_0}=J_{P_{sd}}$.

Step17: By Pt, compute $pr_l(bn+1)$, $bn=bn+1$, go to step2.

Step18: If $J_{P_0}<\infty$, output Pt_{best} as the problem solution, routing succeeded; otherwise, routing failed. The algorithm ends.

4 Performance Evaluation and Conclusion

Simulations of the proposed QoS unicast routing scheme have been done on NS2 (Network Simulator 2) and simulation results have shown that it is both feasible and effective with better performance [6]. In future, our study will focus on improving its practicality, developing its prototype system and extend it to multicast scenario.

References

1. Chuan, X.G., Zi, H.G., Qian, Z., Wen, W.Z.: A seamless and proactive end-to-end mobility solution for roaming across heterogeneous wireless networks. IEEE Journal on Selected Areas in Communications 22(5), 834–848 (2004)
2. Wang, X.W., Hou, M.J., Wang, J.W., Huang, M.: A microeconomics-based fuzzy QoS unicast routing scheme in NGI. In: Yang, L.T., Amamiya, M., Liu, Z., Guo, M., Rammig, F.J. (eds.) EUC 2005. LNCS, vol. 3824, pp. 1055–1064. Springer, Heidelberg (2005)
3. Horst, F.W., Muddassar, F., Yue, Z.: BeeHive: An efficient fault tolerant routing algorithm under high loads inspired by honey bee behavior. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 83–94. Springer, Heidelberg (2004)
4. Yang, L.B.: Principles and Applications of Fuzzy Mathematics, 3rd edn, South China University of Technology Press Guang Zhou (2002)
5. Shi, X.Q.: Game Theory. Shanghai University of Finance Economics Press Shanghai (2000)
6. Liu, C.: Rearch and Simulated Implementaion of Fair Intelligent QoS Routing Mechanism in NGI [D]. Northeastern University, Shenyang (2006)

An Effective Real-Time Rate Control Scheme for Video Codec

Wei Sun and Haoshan Shi

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
sun65488@gmail.com, shilaoshi@nwpu.edu.cn

Abstract. To reduce the complexity of the bit allocation scheme for real-time video codec and keep an acceptable video quality, an effective rate control scheme combined with the fast mode decision is proposed. The length of each GOP(group of pictures) is dynamically determined according to the content of the video clip. The algorithm employs a MAD-tracing based model to allocate bit budget between frames within the same GOP. More motional and active macroblocks(MBs) get more bits than other MBs in the same frame. The median result then can be reused to accelerate fast mode decision procedure. Experimental results show the proposed methods can enhance the overall quality of compressed video.

Keywords: rate control (RC), dynamic GOP length, fast mode decision, video compression.

1 Introduction

In recent years, with the development of the wireless communication system and the rapidly growing demands for video services, several video coding standards have been established (e.g., MPEG-2, MPEG-4, H.264/AVC, AVS). Rate control has always been the central piece especially in the real-time video transmission due to the timing constraint and network bandwidth variation. The bit budget has to be allocated suitably in video sequences to keep reasonable video quality. Rate control includes two steps: bit allocation and bit allocation achievement (which is usually implemented by quantization parameter (QP) adjustment).

TM5[1] in MPEG-2 is a benchmark in rate control scheme, in which the size of GOP is fixed and a constant bit allocation is executed among GOP. GOP has usually been the basic unit for bits allocation. Some subsequent solutions, such as MPEG-4 Annex L rate control and TMN8 [2], also share the same idea and make some amendments (e.g., more accurate bit estimation, adapted with network conditions). TMN8 makes the assumption that the video sequence is nearly stationary in the same GOP and all GOP have similar statistic characteristics, which is not always right in practical applications where the motions in different frames may change dramatically especially when scene-change happened.

Further work has been done in the following ways. First, different bit allocation schemes within GOP scope have been developed with the idea that high-motional frames get more bits and vice versa [3]. Others use new rate-distortion(R-D) models

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 460–467, 2007.

to change some parameters in quantization procedure like QP and λ to achieve an optimized bit allocation [4] [5] [6]. Some annoying quality fluctuation may happen if bit budget were only allocated within fixed-GOP. Dynamic GOP technique has been studied in [7] [10]. Many of aforementioned algorithms are not suitable for real-time video communication due to the large computational complexity.

In this paper, an effective rate control algorithm based on JVT-G012 [8] combined with fast mode decision is proposed for real-time video communication.

2 Mode Decision and MAD Reuse

The mode decision in H.264 is on the basis of eq. (1):

$$J(s,c,MODE,\lambda_{MODE}) = SSD(s,c,MODE) + \lambda_{MODE} \cdot R(s,c,MODE), \tag{1}$$

where s and c are the original blocks and rebuilt ones respectively, λ_{MODE} is the Lagrange multiplier. The SSD(Sum of squared Differences)is computed as

$$SSD = \sum_{i,j} [s(i,j) - c(i,j)]^2 \tag{2}$$

The term MODE includes all of the possible modes for a block to choose. The original full-search of all modes is time-consuming and some fast mode decision schemes have been proposed like [11][12]. Methods like [13] choose to check the MAD(mean absolute difference) or MSE(mean square error) of current block to make a better prediction of best mode. At the same time, rate control schemes have to predict the MAD change in video clips through several prediction models(like linear model in G012) to allocate bit budget. If a sufficient reuse of MAD were executed, not only the bit allocation is more accurate, but the mode decision is accelerated.

2.1 A New MAD Computation Method

A new MAD calculation method is done based on eq. (3):

$$MAD(i,mv(\Delta x,\Delta y)) = \frac{1}{W*H}\sum_{x=1}^{W}\sum_{y=1}^{H} | p_i(x,y) - p_{i-1}(x-\Delta x, y-\Delta y) |, \tag{3}$$

in which $p_i(x,y)$ represents the pixel value of current block, $mv(\Delta x,\Delta y)$ and $p_{i-1}(x-\Delta x, y-\Delta y)$ are motion vector and pixel value of corresponding blocks in the previous picture respectively. W and H are the size of current block which is determined by the situation in previous block. Fig. 1 shows an example of the decision procedure for the shape of current block.

The block-size decision of current block starts from the top-left pixel of the picture. Through the motion vector of previous picture, the top-left pixel finds itself the corresponding pixel in the previous picture and the size of block in which its corresponding pixel belongs is duplicated. So the shape of current block is chosen and should overlap with its corresponding ones. If some pixels is “lost” in corresponding block (this

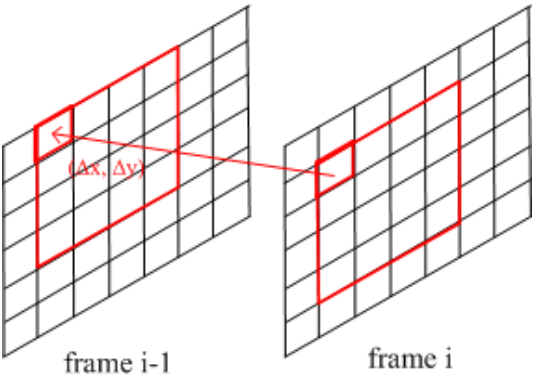


Fig. 1. The decision of current block's shape

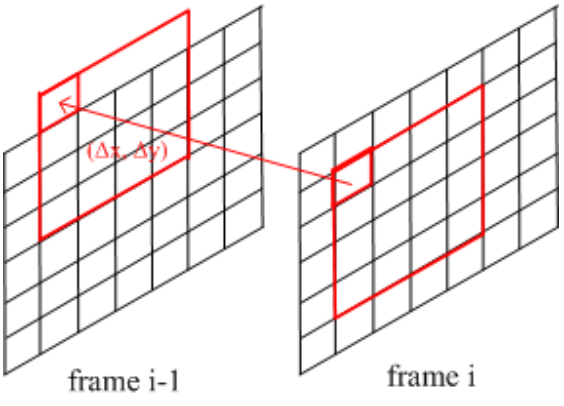


Fig. 2. Lost pixels in previous picture

always happens at the boundary of pictures), as in Fig.2. The MAD of un-overlapping part is calculated according eq. (4), plus to the overlapping part. $p_i(x, y)$ is the pixel value of un-overlapping part. ΔW and ΔH are width and height of un-overlapping part.

$$MAD(i,m,n)_{un} = \frac{1}{\Delta W * \Delta H} \sum_{x=1}^{\Delta W} \sum_{y=1}^{\Delta H} | p_i(x,y) | \tag{4}$$

3 Proposed Rate Control Scheme

The GOP-length in traditional video coding standards is usually fixed, which works fine in typical test sequences like "foreman" and "stefan". However PSNR may change dramatically at low bit rates for non-typical sequences (NTS) where scene-change may happen randomly in a GOP. A NTS is generated by inserting sequence

Fig. 3. Simulation using the NTS with an input frame rate of 30 fps, an encoded frame rate of 15 fps, coded by AVC reference software JM10.1

"foreman" into sequence "stefan" irregularly. Fig. 3 illustrates the impact of fixed GOP on the new test sequence. The visual quality of decoded video drops greatly when scene-change happens within a GOP. Using dynamic GOP length can alleviate the NTS problem. In our experiments, GOP structure is like: I, P, P, P, P……, with none B frames used, which is preferable in real-time video encoding.

3.1 Dynamic GOP Length Control

In section 2, the MAD of every partitioned block in current picture have been calculated and stored. Then the scene-change indicator $Thre(i)$ is decided as follows:

$Thre(i) = 0$

for (m=0; ; m++)

if $MAD(i,m) > Weight1 * (aMAD(i-1)_{average} + b)$

then $Thre(i)$ ++;

"a" and "b" are the two parameters in the MAD linear model of G012 which is updated after encoding every frame. $MAD_{average}(i)$ is defined as follows:

$$MAD_{average}(i) = \frac{1}{M}\sum_{m=1}^{M} MAD(i,m)\,, \tag{5}$$

where M is the number of blocks worked out in section 2. A scene-change may happen if the following condition is satisfied:

$$Thre(i) > 0.75 * M\ . \tag{6}$$

Then buffer occupancy of HRD (hypothetical decoder model [8]) is checked. If buffer occupancy is under 25%, another GOP structure is started, and current frame is encoded to I frame. The factor of buffer occupancy prevents network congestion. If the buffer occupancy is beyond 25%, the GOP structure remains unchanged.

3.2 MB Layer Rate Control

Human eyes are more sensitive on the moving objects. The more motional region deserves more bit budget than other regions during encoding. The activity factor of each sub-MB $RATIO(m)$ is defined as follows:

$$RATIO(m) = \frac{MAD(i,m)}{MAD_{average}(i)} \tag{7}$$

The following is a brief description of QP-modified scheme:

if $RATIO(m) > 1.3$

$QP(m) = QP_c - 1$;

else if $RATIO(m) < 0.5$

$QP(m) = QP_c + 1$;

QP_c and $QP(m)$ is the frame quantization parameter worked out in frame layer rate control and quantization parameter for the current block.

4 Fast Mode Decision

First, the sequence number of various modes is defined in Table 1.

With stored MAD, the proposed fast mode decision is done as follows:

1. if $MAD(i,m) < Weight2 * QP$
   then $MODE_{pred} = MODE(i-1)$
   and $MODE_{candidate} = \{MODE \mid SN(MODE) < SN(MODE_{pred})\}$
   and SKIP mode is calculated first.
2. if $MAD(i,m) > Weight3 * QP$
   then $MODE_{candidate} = \{MODE \mid SN(MODE) > SN(MODE_{pred})\}$
   and Intra 16*16 mode is calculated first.
3. otherwise
   Mode decision is done according [14].

Table 1. Mode and Sequence Number (SN)

MODE	SKIP	16*16	16*8	8*16	8*8	8*4
SN	0	1	2(1)	2(2)	3	4(1)
MODE	4*8	4*4	INTRA 4*4		INTRA 16*16	
SN	4(2)	5	6		7	

5 Experimental Results

Experiments have been conducted to evaluate the proposed rate control scheme compared with G012 using JVT reference software JM10.1 [9]. The test sequence is aforementioned NTS with following parameter settings: QCIF (4:2:0) format, an encoded frame rate of 15 fps. The PSNR result is shown in Fig.4. A better average PSNR (up to 1.1dB) performance is achieved as shown. Fig.5 is the picture of two co-located frames in Fig.3 using proposed scheme. The subjective quality improves a lot than G012.

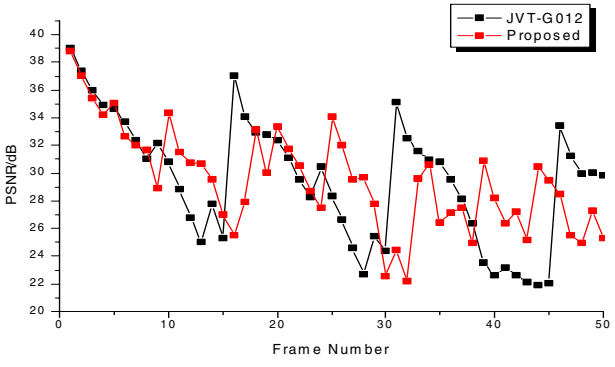


Fig. 4. PSNR of each frame encoding with G012 and proposed rate control scheme

Fig. 5. Pictures of the co-located frame in Fig. 3 using proposed rate control scheme

To verify the performance of revised fast mode decision scheme, four typical QCIF image sequences are selected out to be compared here, which are “akiyo”, “foreman”, “Stefan” and “coastguard”. “akiyo” is a slow-speed sequence, and “foreman” is a middle-speed one. “Stefan” and “coastguard” are fast-speed sequences. Each of them has 100 frames, with GOP structure of IPPPP……. The sequences are encoded at various conditions with rate control. To simplify comparison, we have used average PSNR gain (ΔPSNR), bit-rate reduction (Δbitrate) and time reduction (Δtime) which are defined as follows:

$$\Delta\text{PSNR} = PSNR(Y)_{proposed} - PSNR(Y)_{ref}$$

$$\Delta bitrate = \frac{bitrate_{proposed} - bitrate_{ref}}{bitrate_{proposed}} * 100\%$$

$$\Delta time = \frac{time_{proposed} - time_{ref}}{time_{proposed}} * 100\%$$

$time_{proposed}$ and $time_{ref}$ are the time spend on mode decision using proposed rate control scheme and G012 scheme respectively(other definition are the same).

Table 2. Performance of proposed scheme in 32Kbps and 15frames/s

Sequence	Akiyo	Foreman	Stefan	Coastguard
ΔPSNR	0.2	-0.2	0.05	0.1
Δbitrate(%)	2.21	-3.61	6.03	3.09
Δtime(%)	-4	-8	-16	-12

Table 3. Performance of proposed scheme in 64Kbps and 30frames/s

Sequence	Akiyo	Foreman	Stefan	Coastguard
ΔPSNR	-0.4	0.01	0.03	0.04
Δbitrate(%)	4.21	-1.22	1.23	1.07
Δtime(%)	-6	-11	-14	-11

Table 4. Performance of proposed scheme in 128Kbps and 30frames/s

Sequence	Akiyo	Foreman	Stefan	Coastguard
ΔPSNR	-0.3	0.07	0.09	0.03
Δbitrate(%)	1.67	-2.37	1.98	3.01
Δtime(%)	-4	-6	-11	-9

From Table 2-4, we can observe that the most decreased PSNR is 0.4 dB and the most increased bit-rate is 6.03% when compared with the reference algorithm in [14]. In some cases, the PSNR increases and the bit-rate decreases. The whole decreased encoding time in mode decision changes with the sequence of video, the least decreased time is 4% and the most decreased time is 16%. We can also observe clearly that the acceleration of proposed algorithm has direct relationship with the characteristic of the video sequences: the acceleration of sequences with small motion vectors is lower than that of sequences with big motion vectors.

6 Conclusion

This paper presents an improved and effective rate control scheme for real-time video combined with MAD reuse. The experimental results show the proposed methods get better PSNR quality and better subjective quality compared with the AVC rate control

schemes. And fast mode decision is speed up by up to 16% than the referential algorithm. Further work will be done on self-adjustment of several parameters in the scheme.

References

1. MPEG-2 Test Model 5,Doc. ISO/IEC JTC1/SC29 WG11/93-400 (1993)
2. ITU-T, Video codec test model, near-term, version 8 (TMN8), H.263 AdHoc Group, Portland (1997)
3. Xie, B., Zeng, W.: A Sequence-Based Rate Control Framework for Consistent Quality Real-Time Video. IEEE Trans. Circuits Syst. Video Technol. 16, 56–71 (2006)
4. Jiang, M., Ling, N.: On Lagrange Multiplier and Quantizer Adjustment for H. IEEE Trans. Circuits Syst. Video Technol. 16, 663–669 (2006)
5. Yuan, W., Lin, S.: Optimum Bit Allocation and Rate Control for H.264/AVC. IEEE Trans. Circuits Syst. Video Technol. 16, 705–715 (2006)
6. He, Z., Kim, Y.K., Mitra, S.K.: Low-delay rate control and smoothing for video coding via p-domain source modeling. IEEE Trans. Circuits Syst. Video Technol. 11, 928–940 (2001)
7. Gu, X., Zhang, H.: Implementing Dynamic GOP in Video Coding, Multimedia and Expo, 2003. In: ICME 2003, Proceedings, vol. 1, pp. 349–352 (2003)
8. Li, Z., Pan, F.: Adaptive Basic Unit Layer Rate Control for JVT, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Pattaya II, Document JVT-G012, Thailand (2003)
9. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG Reference Software JM 10.1
10. Li, H., Liu, G.: Adaptive scene-detection algorithm for VBR video stream, IEEE Trans. Multimedia 16, 624–633 (2004)
11. Jeon, B.: Fast mode decision for H.264, JVT-J033, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6) 10th Meeting: Waikoloa, Hawaii (2003)
12. Chang, A., Au, O.C., Yeung, Y.M.: A novel approach to fast multi-block motion estimation for H.264 Video Coding. In: ICME 2003. Proc. of IEEE Int. Conf. on Multimedia & Expo (2003)
13. Chen, J., He, Y.: A fast mode decision algorithm in H.264. In: Proceedings of PCS 2004, San Francisco, p. 46 (2004)
14. Tourapis, H.C., Tourapis, A.: Fast motion estimation within H.264 codec. In: ICME 2003, Baltimore, Maryland, USA (2003)

An Anti-statistical Analysis LSB Steganography Incorporating Extended Cat-Mapping

Wenxiao Chen, Jing Cai, and Siwei Li

Department of Communication and Information Engineering
Nanjing University of Posts and Telecommunications P.R. China
njzykvsh@163.com
caijinghere@gmail.com
lisiwei5555@sina.com

Abstract. In this paper, we propose a modified LSB steganography which resists most of the popular statistical analysis-based steganalysis,such as SPA(Sample Pair Analysis) and RS. The modified algorithm does not solely depends on embedding secret message into the cover image as the conventional LSB watermarking method does, but increases or decreases the candidate pixel's value by two in accordance with the number of the pixel's surrounding pixels whose LSB is positive. We discuss on the basic principle of the proposed algorithm and experiments show that the improved LSB steganography has a good robustness to statistical analysis. An extended cat-mapping is introduced in this paper which will better encrypt the secret message than the standard cat mapping. Experiments show the histogram of the encrypted image will be flat and resembles the white noise's histogram, which significantly enhances the security of the algorithm.

1 Introduction

Steganography is a hot research subject in the field of information security. It is an art of secret communication[1] and makes the secret communication available by embedding messages into cover objects(objects not containing any secret message).It is essential that the stego object(objects containing the secret message) does not contain any detectable artifacts that could be detected by an observer. With the proliferation of digital images and given the high degree of bit redundancy present in the image, an increasingly rising attention is given to digital image and digital images are used as the cover image for steganography. In the past years, a lot of researches have been done and some effective watermarking methods have been developed. Among them, LSB (Least Significant Bit) steganography is one of the simplest watermarking algorithms with high embedding ratio of secret information. It is a simple approach to embed messages into a cover image.The major advantages of LSB steganography are its simplicity and high watermarking bit ratio.However, with the deepening of the steganalysis research, there emerge some steganalysis methods which can effectively detect LSB steganography. Fridrich developed the RS steganalysis. This

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 468–476, 2007.

method makes statistical analysis of the alterations of regular groups and singular groups in the image to estimate the length of the message embedded. Sorina Dumitrescu proposed SPA, another steganalysis to detect LSB steganography via sample pair analysis. When the embedding ratio is more than 3%. Generally speaking, RS and SPA steganalysis are the two most reliable detection methods of LSB steganography until now[8]. If we look through these steganalysis, we will find that both RS and SPA detecting methods generally take advantage of the relation between the image's LSB plane before and after the standard LSB embedding, analyze the statistical difference and derive out the length of the embedded message. This sort of statistical analysis is often fatal to the traditional LSB information hiding scheme, which solely depends on embedding message into the least significant bit of an image.

However, we see most of the steganalysis mentioned above are constructed on the basis of some statistical prerequisites. If we destroy these statistical analysis prerequisites, adjust the embedding algorithm, then the stego will no longer be so easily detected by such statistical analysis-based steganography. In effort to achieve this goal, an improved LSB steganography algorithm which is reliably robust to such statistical detection is proposed in this paper. The algorithm increases or decreases the candidate pixel's value by two in accordance with the number of the pixel's surrounding pixels whose LSB is positive. Actually, it is embedding message into the second LSB layer of the image. Next section,we will describe this algorithm in detail.

2 Proposed LSB Steganography

Suppose, s denotes one secret bit to be embedded, u denotes the corresponding pixels whose 2nd LSB layer is to carry the secret bit s. Let v be the pixel which has already been embedded with s. We use $|u|$ to denote the value of the pixel u, and $|v|$ to denote the value of pixel v. The position of the corresponding pixel is selected by another algorithm which requires a key k to control the generation of the candidate pixels's position in the cover image. Let's discuss this later. The existence of key k adds to security of the watermarking system.

The proposed LSB algorithm goes like this: If the current secret bit s, zero or one, is identical to the 2nd LSB of the current candidate pixel's value $|u|$, the pixel keeps its original value. If not, check whether the value of current pixel is 255 or 0. If 255, decrease the value by two. If 0, set the value to be 2. If $|u|$ equals to neither 255 nor 0, the proposed algorithm will modify the value of pixel u by the checking the number of its surrounding pixels whose LSB is 1. Pixel u's 4 surrounding pixels are those pixels which are nearest to u and are on the left-hand side, right-hand side, up-hand side, down-hand side of pixel u.Figure.1 shows pixel $u's$(black) 4 surrounding pixels(gray).

If the candidate pixel u is on the edge of the cover image, then we only count its available surrounding pixels. The algorithm will check these 4 pixels. If there is an even number of surrounding pixels whose LSB is 1, decrease $|u|$ by two. If there is an odd number of surrounding pixels whose LSB is 1, increase $|u|$ by two.

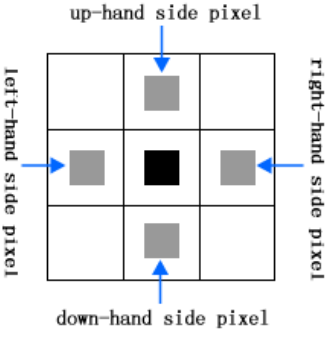


Fig. 1. Four surrounding pixels of a given pixel

2.1 Basic Principle of Proposed Algorithm

As a matter of fact, the proposed algorithm is to randomly increase or decrease the candidate pixel's value in a sense. The value change of the candidate pixel is determined by the number of its surrounding pixels whose LSB is positive. Therefore it is independent on the secret bit information. From the statistical point of view, for a particular pixel u, the number of its surrounding pixels who has a positive LSB is random and the possibility is even.

We still use $|u|$ to denote the candidate pixel's value which ranges from 0 to 255. Let's denote the number of pixels which has an odd pixel value (1,3...255) by N_{even} and denote the number of pixels which has an even pixel value (0,2...254) by N_{even}. For a given image, the possibility for an even-valued and an odd-valued pixel to appear are almost identical. Let's denote the possibility of even-valued pixel to appear by P_{even} and denote the possibility of odd-valued pixel by P_{odd}. From the statistical perspective, $P_{odd} = P_{even}$.

If we embed a secret bit into a candidate pixel, we first check its surrounding pixels. However, no matter whether we increase or decrease the candidate pixel's value, both N_{even} and N_{even} will remain unchanged because we increase or decrease by two instead of one. Thus we still have the equation : $P_{odd} = P_{even}$.

Experiments show that the proposed algorithm will not generate statistical difference from the original image as the conventional LSB algorithm does. The results also indicate the modified LSB algorithm has an embedding ratio as high as the traditional algorithm. The degradation it brings to the image is just the same as the standard LSB coding method does.

2.2 Embedding Process

A pre-processing of the watermark can be performed in effort to improve security. Before embedding the secret message into the cover image, we first the transform the message via Arnold transformation.

In this paper, we extend the standard Arnold transformation to encrypt the secret watermark. Experiments show that the extended Arnold transformation

will significantly change the histogram of the encrypted image and will turn it into a chaotic state which resembles the white-noise. Even if the intruders detect out the message, it is just like a Gauss noise that is useless. This feature largely enhances the security of the encrypted watermark image. We will see this later.

2.3 Pre-processing of Watermark

The 2-dimensional Arnold transformation, usually called cat mapping is to transform one matrix into another. It can be viewed as a discrete chaotic system. Without loss of generality,let M be a $N \times N$ matrix, the element (i, j) can be shifted to another position $(i'j')$ in the matrix by

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = P \begin{bmatrix} i \\ j \end{bmatrix} \bmod N$$

$$\text{where } P = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

where mod N represents the modulo operation and (i, j) represents a particular pixel's coordinates in the image and so does $(i'j')$. In order to guarantee the cat mapping is a one-to-one mapping, the value of the matrix P should meet the requirement of $|P| = 1$. So, the matrix P can be simply denoted by

$$P = \begin{bmatrix} 1 & a \\ b & 1 + ab \end{bmatrix},$$

where both a and b are integers. Therefore the Arnold transformation can be written as

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 + ab \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \bmod N$$

The Arnold transformation or cat mapping is widely used as a method to encrypt image in the field of watermarking. The basic principle of the cat mapping is to rearrange the location of pixels within the image. It achieves its objectives of encryption by disturbing the position of pixels. But experiments shows that cat mapping could not change the histogram of the encrypted image thus providing opportunities for observers to speculate the original image. In this paper, we extend the cat mapping by taking into account the pixel value but not only the coordinates of the pixel.

The extended cat mapping is defined as following:

$$\begin{bmatrix} i' \\ j' \\ v' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 + ab \\ p & q \end{bmatrix} \begin{bmatrix} i \\ j \\ v \end{bmatrix} \bmod N,$$

where v represents the pixel's value and v' represents the modified pixel value. Here both p and q are integers.

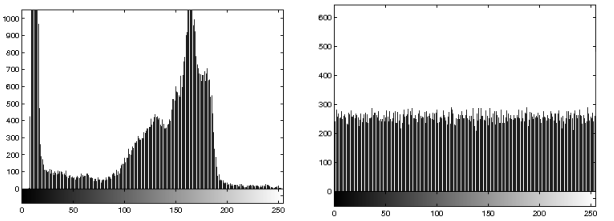


Fig. 2. Histograms of images via standard cat mapping and extended cat mapping

We can see the extended cat map changes the pixel's value while rearranging the coordinates of the pixel. Experiments show that the histogram of the encrypted image has white-noise characteristics. Figure.2 shows this feature. It is easy to see that the the second part of fig.2 which displays the histogram of the transformed image via the extended cat mapping is more flat and resembles pseudo-random sequence's histogram.

The reverse transformation is also easy to implement. It requires the parameters of a, b, p, q as the key of the watermark encryption system. Even if the intruders successfully detect out the embedded watermark, without the valid key, it is only a pseudo-random sequence which is useless to the intruders, therefore significantly enhances the watermarking system's security.

2.4 Embedding Procedure

Step1. Transform the watermark image via the Extended Arnold Transformation Equation.

Step2. Select the positions to embed message with the initial user's key k.

Step3. Check whether the cover image is a color image or a gray-scale image. If it is a gray-scale image, embed the message to the only color space. If it is a color image, separate it into different color spaces and embed secret message into the color spaces respectively.

Step4. We first embed the message length n into the cover image I. This information tells the receiver how many bits should be extracted from the stego image. The mount of pixels required to store can be got by $\lfloor \log_2 n \rfloor + 1$.

Step5. Select a candidate pixel u to embed the current message bit s. Use the above algorithm to modify the pixel value $|u|$.

Step6. Select next candidate pixel and repeat the operation of Step5 until all message are embedded. Now, we get the stego image I'.

Figure.3 shows the proposed 2nd LSB layer embedding procedure.

3 Extraction Process

Actually,extraction is the reverse process of the embedding process. We first extract the 2nd least significant bit of the candidate pixels and then transform the

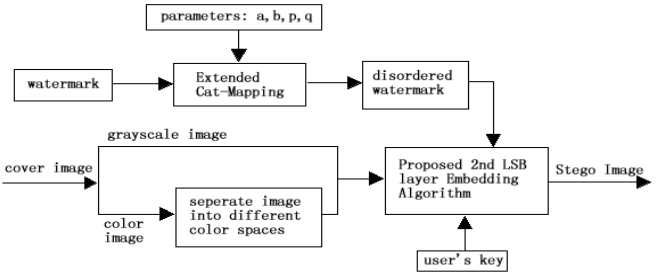


Fig. 3. Proposed 2nd LSB layer embedding procedure

disordered watermark via the extended cat mapping.The extraction is relatively simple and is not the focus of this paper, so we do not give a lengthy description here.

4 Experimental Results

In order to test the proposed algorithm's robustness against statistical analysis, we choose a host of standard test images to embed messages. The types of message we embedded include a grayscale 'copyright' bitmap as well as some random pseudo sequence generated by Matlab. Figure.4 are some of the test images and message image we chose.

For the purpose of compassion, all these cover images are watermarked by the standard LSB method and the proposed algorithm with the same embedding ratio. Figure.5 shows the histograms of the stego Lena image watermarked by

Fig. 4. Some of the test images and copyright message

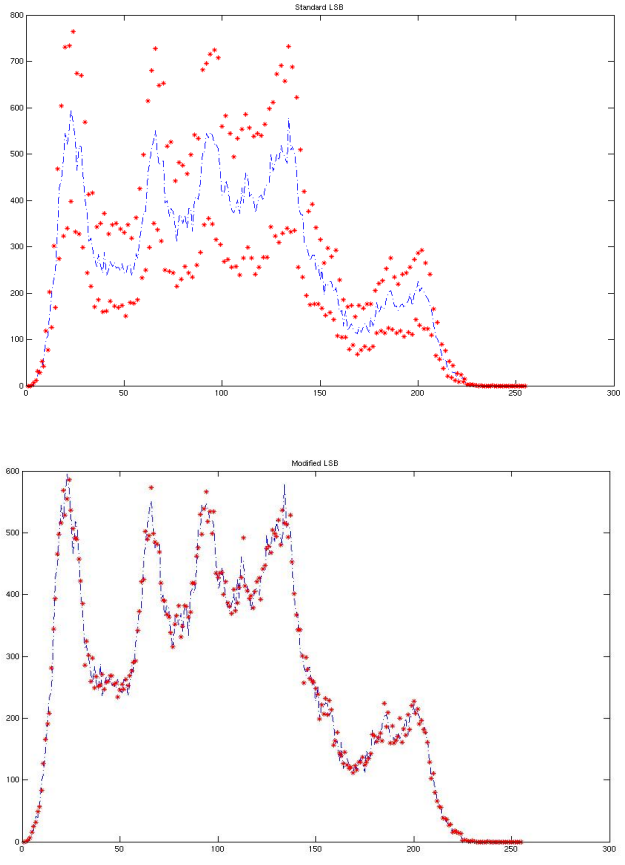


Fig. 5. Histograms of stego image with Standard LSB And Modified LSB

standard LSB method and the proposed LSB algorithm respectively. From the Fig.5, we see that when using the traditional LSB coding method, the histogram of the stego image(which is shown by the red dots) departs from the original image's histogram (which is shown by the blue line) significantly. It is because in the areas where the original bit does not equals to the watermark bit, the standard the LSB coding method produces a constant error. This always rising error leads to the difference between the histograms of the cover image and the stego image. It obviously provides opportunities for attackers to analyze and detect the message embedded via histograming. However, when applying the proposed algorithm, the error produced is minimized, nearly zero. The improved algorithm decreases or increases the pixel value in accordance with the number of the current pixel's surrounding pixels whose LSB is positive thus from the statistical point of view, the possibility to increase and decrease is even ,therefore it produces a near-zero error in the histogram. When applying the proposed embedding method, the histogram of the stego image is almost identical to that of

the original image. It leads the statistical analysis attackers to make an incorrect judgment thus, to a large extent, enhances the robustness of the watermark.

4.1 Robustness Against RS and SPA

We implemented the proposed algorithm with 120 various types of sample images of BMP format. These images are embedded with 10%, 40%, 70% secret messages respectively. Table.1 shows the experimental results of resisting RS detection and SPA detection. We see, with the uniform detecting threshold of 3%, both RS and SPA test could not detect most of the stego images.

Table 1. RS and SPA Detection Results

Message Embedding Ratio	10%	40%	70%
RS(threshold of 3%)	0/120	0/120	1/120
SPA(threshold of 3%)	0/120	1/120	2/120

4.2 Degradation of PSNR

The Peak Signal to Noise Ratio (PSNR) is an important measurement to gauge the degradation which is brought to the image by watermarking. It reveals the difference between the original image and the stego image. The difference declines as the PSNR gets larger. Typically, the Peak Signal to Noise Ratio is defined as

$$PSNR = 10lg(\frac{MAX(x_{ij})^2}{MSE})$$

where M and N are the length and width of the cover image respectively,and MSE is the mean-square error between stego or an attacked stego image and the original image. x_{ij} denotes the pixel value of the cover image and $x_{ij}^{'}$ denotes the pixel value of the stego image. Table.2 shows the PSNR of the stego image when using the 'copyright' message and random pseudo sequence of different embedding ratio as the embedding message.

Table 2. PSNR of different embedding ratio

Embedding Ratio	Copyright	10%	40%	70%
Standard LSB PSNR	58.3%	64.2%	55.7%	56.2%
Improved LSB PSNR	54.3%	60.2%	51.7%	50.2%

We see that the proposed 2nd LSB layer embedding method does not introduce large PSNR degradation to the image compared with the standard LSB steganography while improving the watermark's robustness to statistical analysis significantly.

5 Conclusion

This paper presents a modified LSB Steganography which will defeat most of the popular statistical analysis-based Steganalysis,such as SPA and RS. The modified algorithm increases or decreases the candidate pixel's value by two in accordance with the number of the pixel's surrounding pixels whose LSB is positive. experiments show that the proposed LSB Steganography has a good robustness to statistical analysis.We also introduced an extended cat-mapping in effort to enhances the security of the algorithm.

Acknowledgement

This research work is supported by the STITP foundation program (No.070020612) of Nanjing University of Post and Telecommunication(P.R. China).

References

1. Fridrich, J., Goljan, M., Du, R.: Detecting LSB steganography in color and gray-scale image. Magazine of IEEE Multimedia, 22–28 (2001)
2. Fridrich, J., Goljan, M., Du, R.: Reliable detection of LSB steganography in color and grayscale images. In: Proc. ACM Workshop Multimedia Security, Ottawa, ON, Canada, pp. 27–30 (October 2001)
3. Dumitrescu, S., Xiaolin, W., Wang, Z.: Detection of LSB Steganography via Sample Pair Analysis. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 355–372. Springer, Heidelberg (2003)
4. Zhang, T., Ping, X.: A New Approach to Reliable Detection of LSB Steganography in Natural Images. Signal Processing 83(10), 2085–2093 (2003)
5. Westfeld, A.: Detecting low embedding rates. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 324–339. Springer, Heidelberg (2003)
6. Fridrich, J., Du, R., Meng, L.: Steganalysis of LSB Encoding in Color Images. In: Proc. IEEE Intl. Conf.Multimedia and Expo, CD-ROM, IEEE Press, Piscataway, N.J (2000)
7. Yu, J.J., Han, J.W., et al.: A Secure Steganographic Scheme against Statistical Analyses. In: Kalker, T., Cox, I., Ro, Y.M. (eds.) IWDW 2003. LNCS, vol. 2939, pp. 497–507. Springer, Heidelberg (2004)
8. Provos, N.: Defending Against Statistical Steganalysis. In: 10th USENIX Security Symposium, Washington, DC (2001)
9. Westfeld, A. Tzmann, A.P.: Attacks on Steganographic Systems[C]. In: Proc. of 3rd International Workshop on Information Hidding, Dresden, Germany (1999)

Geographic Probabilistic Routing Protocol for Wireless Mesh Network

Ning Xiao, Ling Ding, Minglu Li, and Minyou Wu

Computer Science Department, Shanghai Jiao Tong University, Shanghai 200240, China
{xiaoning,dingling,mlli,mwu}@sjtu.edu.cn

Abstract. This paper presents GPR (Geographic Probabilistic Routing protocol), an opportunistic routing protocol worked between mesh routers in WMNs (Wireless Mesh Networks). In GPR, nodes detect the link condition by probe packets. In order to send a packet, the sender selects a candidate subset. The nodes who successfully received the packet send ACK according to their priority. If there are no ACKs from other candidates, instead of sending ACK immediately, the candidate broadcasts ACK and transmits the packet at transmission-probability. The extensive simulation results show that GPR is promising to achieve higher throughput and better scalability compared to the reference method.

1 Introduction

Wireless Mesh Networks (WMNs) are multi-hop wireless networks. An optimal routing protocol for WMNs should select a multi-hop path quickly, avoid traffic congestion, and work well as the network scale increases [1]. The traditional approach to route traffic in WMNs is to adopt shortest path routing schemes which are similar to the methods used in wired networks. While these schemes are effective in wired networks, where a transmission link is either successful or failed; they can not cope with the unreliable or unpredicted medium of wireless networks. Opportunistic routing [2, 3, 4, 5] is a novel research area for wireless networks to deal with the trustless medium. Opportunistic routing broadcasts packets firstly and then chooses the next hop receiver based on which neighbor has successfully received them.

One of the key challenges in opportunistic routing is to maximize the distance a packet advanced in one transmission without causing much duplicate transmissions or incurring significant coordination overhead [3]. A lot of researches make efforts to get over this challenge. Most of these methods piggyback the information about the receiver information to the data packet and diffuse them in the networks [3, 4, 5]. MORE [2] induce network encoding into opportunistic routing to renew packets so there are no duplicate transmissions.

Most of the previous works do not provide an opportunistic routing to work in large scale networks. The reason is that the coordination overhead increases as the network scale does. This paper proposes Geographic Probabilistic Routing (GPR), a novel opportunistic routing works well in large scale networks. Instead of preventing ACK lost, GPR reckons that missing ACK message is inevitable in wireless networks and estimates the probability at which this situation would happen. GPR is based on

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 477–486, 2007.

the geographic routing [13, 14, 15] since the geographic distance a packet moved forward can be simply and clearly weighed. We evaluate GPR through NS-2 simulations with diverse network topologies and make the following findings:

- GPR provides a way for node to decide whether a packet should be forwarded when it does not know which nodes had also received the packet. In simulations, GPR's throughput increases more than 50% at the expense that the duplicate transmissions fluctuate in the range of 0.57~1.36. The throughput declines slowly as the number of simultaneous flows increases while the duplicate transmissions keep in the same area.
- GPR performs well as the network scale grows. Since the coordination overhead does not increase significantly as the network scale grows, GPR can work properly in large scale networks. In our simulations, when averagely packets come through 3 relay nodes to get to the destination, GPR deliveries more than 50% of the total packets successfully.

The rest of the paper is organized as follows. In section 2 we present the related works. We give a simple motivation example in section 3 and describe GPR in details in section 4. We illustrate our simulations and the results in section 5. The conclusion is given in section 6.

2 Related Works

In opportunistic routing, there is an intuitive tradeoff between duplicate transmissions and coordination overhead. We can increase the communication between the sender and receivers to decrease duplicate transmissions, or communicate as little as possible at the expense of multiple retransmissions. Researchers have brought out their way to achieve these purposes.

ExOR [4][5] is the first completely opportunistic routing protocol. In 2003, S. Biswas advanced the primal scheme of ExOR [5] which we called as ExOR-1. In this scheme, candidates send ACK messages in turns and decide whether to forward the packet according to the ACKs it has received. ExOR-1 does not give an effective way to constrained duplicated transmission when the ACKs are not received correctly.

In 2005, S. Biswas brought forward another scheme of ExOR [4] that we named as ExOR-2. It strictly schedules the routers' access to the medium before a batch of packets (10-100 packets per batch) is broadcast by the initial sender. ExOR-2 uses the batch map to record which packets each node has received and diffuses it with the data packets; every relay node only forwards the packets that have not been acknowledged by nodes closer to destination. ExOR-2 effectively decreases duplicated transmissions and provides significant throughput improvement. However, supporting multiple simultaneous flows is still an open question in ExOR-2.

SOAR [3] introduced the priority-based forwarding mechanism to prohibit duplicate transmissions. It spreads ACK message in the network in a cheap way to constrain duplications and improves the throughput significantly. In SOAR, there must be more intra-flow interference for the relay nodes when there are multiple packets transmitted between the same source-destination pair.

MORE [2] combines network encoding and opportunistic routing to support multiple simultaneous flows. Using this method, source node creates random linear combinations of packets and broadcasts the coded packets continually. The relay nodes combine the independent packets and forward them. The destination sends ACK message to source along the shortest path when it can decode the independent packets. Cooperation network coding into opportunistic routing is a novel idea in this area.

3 Motivation Example

In the part, we explain the motivation of GPR. As shown in Fig.1, source *S* has a packet for destination *D*. Three nodes relay the packet for *S*, and does it in this turn: node 1, node 2 and node 3. The direction of links shows the transmission direction and the transmission reliabilities are labeled above them. Every node knows their forward reliability and backward reliability with its neighbors. *S* sends the packet containing the candidate subset and the corresponding forwarding reliabilities. Received the packet, nodes 1-3 send ACK messages along the following process:

(1) Node 1 receives the packet successfully, it sends ACK message and forward the packet; other nodes drop the packet if they receive the ACK message.

(2) Node 2 receives the packet and waits for node 1's ACK message. If it does not hear the ACK, node 2 need to decide whether to send ACK itself. There are mainly two reasons that node 2 does not receive ACK message from node 1: first, node 1 does not receive the data packet, the probability this situation happens is: $1-0.3 = 0.7$; secondly, node 1 successfully receives the packet and sends ACK, but node 2 missed it and this happens at the probability: $0.3\times(1-0.7) = 0.09$. So node 2 send ACK at probability: $0.7 / (0.7+0.09) = 0.886$.

(3) Node 3 waits for the other two nodes' messages to decide whether to forward the packet. Like node 2, it calculates the probability to send ACK if no message have been heard. Node 3 transmits the packet when it does not hear any message from other nodes at probability: $(0.7+0.5) / (0.7+0.12+0.5+0.25) = 0.764$.

If source *S* heard ACK message from any of the candidates, it drop the duplicate for the packet. We will explain GPR including nodes in detains in the next section.

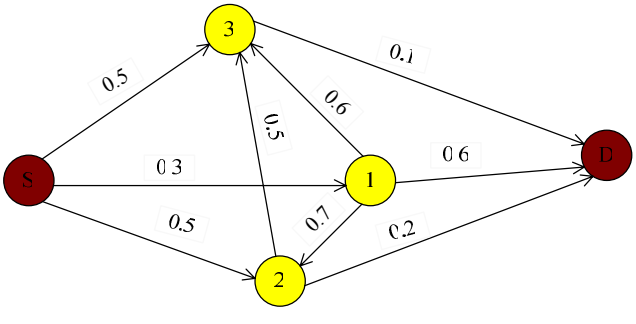


Fig. 1. The motivation example of GPR

4 Geographic Probabilistic Routing Protocol

As explained above, in GPR the candidates estimate the probability to forward a packet if it does not hear ACK from other candidates. Since it chooses the candidate set based on the neighbors' location information, GPR works in the following scene:

At first, every node in GPR is static. This is because lots of mesh routers are deployed and not moved again in the WMNs. Secondly, the network is dense enough that each node can get the candidate nodes. Thirdly, the nodes know the topology of the network. This can be achieved by using GPS devices and so on.

GPR cooperates with extended 802.11 mac protocol to achieve probabilistic forwarding. Extended 802.11 mac protocol listens to the mac layer ACKs, calculates the transmission-probability and retransmits the packets. In the following sections, we describe the GPR and extended mac protocol in detail.

4.1 Communication Condition Estimation

At first, we give the definition of the forward reliability and the backward reliability; then explain how to get them.

- $P_{ij}(i)$: the forward reliability from node i to its neighbor node j which is knew by node i;
- $P_{ji}(i)$: the backward reliability from its neighbor node j to node i itself which is knew by node i.

GPR can be divided into information gathering period and packet transmission period. After the deployment of mesh routers, nodes begin to gather the initial information about the link property. During this phase every node sends probe packet per second. The packets from node i contain the list of its neighbors and the corresponding backward reliability $P_{ji}(i)$ (node j is node i's neighbor). Node j received the probe packets does two things: firstly, it updates its backward reliability $P_{ij}(j)$; then, if the probe packet contains the node backward reliability $P_{ji}(i)$, node j renews its forward reliability $P_{ji}(j)$ with it. Given the beaconing frequency is known, these reliabilities are measured by the number of messages received successfully in the last 10 seconds over the number of messages expected to be received during this time interval.

GPR uses EWMA (Exponentially Weighted Moving Average) [3] as the link estimator. EWMA is a linear combination of infinite history with exponential weights. Let $\overline{P}_t$ be the current reliability estimation, n is the number of known missed packets, m is the number of received packets, and w is the window size. Then the reliability is updated as (1):

$$\overline{P}_{t+1} = \overline{P}_t \times \alpha + P_{t+1} \times (1-\alpha) \tag{1}$$

where $0 < \alpha < 1$, $P_{t+1} = \dfrac{m}{m+n}$ and m and n are reset to 0 when $m+n > w$.

When node i gets its forward reliability $P_{ij}(i)$ and backward reliability $P_{ji}(i)$ with neighbor j, the ETX_{ij} (i) (Expected Transmission Times)[6] between these two nodes is:

$$ETX_{ij}(i) = \frac{1}{P_{ij}(i) \times P_{ji}(i)} \tag{2}$$

After the information gathering period, every node has constructed its neighbor table. The table contains the nodes it hears directly and the corresponding estimated link reliabilities. When the node has a packet to send, it chooses the nodes from this table to form the candidate subset.

During the packets transmission period, the probe packets are still sent, so the neighbor tables contain the most recent link information. And, if a node has not been heard for a predefined time, we judge it as failed. Nodes update their neighbor tables and clear the failed neighbor's entry, thus there would be no packet sent to it.

4.2 Candidate Subset

As mentioned at the beginning of section 4, the nodes in the network keep the topology map of the network. When a node has packets to send, its can get the location of the destination and its neighbors, then uses these information to get the candidate subset. The routing algorithm will be explained in detailed using Fig. 2.

In Fig.2, nodes 1-7 are the neighbors of source *S*. When *S* wants to send packets to destination *D*, it looks up in the neighbor table, and selects the neighbors that fall within an angle θ in the direction of *D*. The value of θ is picked big enough to guarantee that there are nodes fall into this area in a dense network, but must be smaller than 180°. Thus no packet is sent in the backward direction. To ensure packets do not cycle in a loop, every time the packet is forwarded, it must never be send to nodes farther than the destination. If the packet is received by a node projected beyond the direct line between source-destination pair, it is dropped.

There may be a lot of nodes can be chosen as relay nodes, and some of them have very low transmission reliabilities. We choose the relay nodes by *ETD* (Expected Transport Distance) calculated by (3):

$$ETD_{si} = Dist_{si} * \cos\alpha / ETX_{si}(s) \tag{3}$$

As showed in the Fig.3, α is the relative inclination between the line from *S* to *D* and the line from *S* to the intermediate node *i*. $Dist_{si}$ is the distance between source *S* and node *i*. Thus $Dist_{si}$*cosα is the effective forward distance from *S* to node *i* in the direction of the destination. *ETD* is the expected transmission distance between *S* and node *i* over a time interval. In other words, GPR chooses the relay node which can make the largest expected positive forward distance.

Once get the candidate subset, *S* sorts the candidate nodes in the decreasing order of their effective forward distance in the direction of *D*. So a candidate node making the largest forward distance in direction of destination *D* sends ACK firstly. The *i*-th forwarding node in the list sending their ACK messages after $(i-1)\times\delta$. δ is the time to listens to ACK message from a higher prior node. To limit the delay variance and reduce overhead, we limit the maximum number of forwarding nodes to 4.

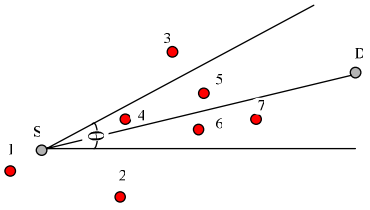


Fig. 2. A simple topology

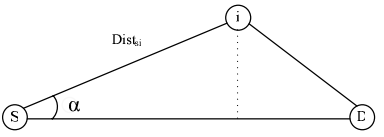


Fig. 3. Example of ETD

4.3 Probabilistic Forwarding

In Fig.2, nodes 4-7 compose the candidate subset for the packet. Source *S* adds the forwarding list contains nodes' id and the forward reliabilities $P_{si}(s)$ ($i = 4\ldots7$) in the packet header. The priority of the forwarding nodes is demonstrated by their position in the list.

When a node receives this packet, first of all, it checks if it is in the forwarding list: the nodes do not be contained in the list just drop the packet; the nodes in it send ACK message orderly. As the example in section 3, if a node receives the ACK messages from the others, it drops the packet. Otherwise it estimates the transmission-probability. If the low priority node does not receive ACK from the high priority ones, there are two major reasons: the high priority nodes do not receive the packet; or the high priority ones receive the packet and send ACK, but it has missed them. Since the low priority node sends back ACK only if the high priority ones had missed the packet, the low priority node sends ACK in accordance with (4):

$$\begin{cases} T_j = 1 & (j = 1) \\ T_j = \dfrac{\sum_{i=1}^{j-1}(1 - P_{si}(s))}{\sum_{i=1}^{j-1}((1 - P_{si}(s)) + P_{si}(s)(1 - P_{ij}(j)))} & (j > 1) \end{cases} \tag{4}$$

T_j is the transmission-probability of node *j* for the packet. When a node does not receive any ACK message, instead of transmitting the packet immediately, it broadcasts ACK and sends the packet at the possibility calculated by (4).

In the lossy wireless network, there could be no candidates received the packet thus no ACK was heard by the sender. In these cases, GPR uses mac layer retransmission to send the packet again. In order to be compliant with 802.11 mac protocol, the maximum retransmission times is set to 4.

4.4 Decreasing Duplicated Transmission

Because of the unreliable wireless medium, the following undesired situation would happen. One of the relay candidates sends ACK message and forwards the packet. While the other candidate receives this ACK and drops the packet, just the sender misses it and retransmits the packet in vain. This is a major reason for duplicates. In order to decrease this kind of duplications, every relay node remembers the packet it has processed. If it receives a packet more than once, the node only sends the ACK to

inform the others that it received the packet. Instead of forwarding it once again, the relay candidate just drops the packet quietly.

5 Simulation and Evaluation

We simulated our protocol using NS-2 [10] and compared it with the SOAR [3]. In this part we describe the simulation results and evaluation in particular.

5.1 Simulation Environment

Data Link Layer Model: We extended IEEE 802.11 mac protocol and disabled the RTS/CTS to cooperate with GPR. As the baseline comparison, SOAR works with IEEE 802.11 disabled the RTS/CTS [3] as well. In the simulation, we think the transmission reliability only changes with distance. The transmission range of mesh router changes between 70m to 300m randomly in the simulation environment. Table.1 is the transmission reliability at different distances:

Table 1. The transmission reliability at different distances

Distance	70m	80m	90m	100m	110m	120m	130m	140m
Transmission Reliability	1.000	0.883	0.785	0.701	0.628	0.565	0.508	0.457
Distance	150m	160m	170m	180m	190m	200m	210m	220m
Transmission Reliability	0.412	0.370	0.311	0.296	0.263	0.233	0.205	0.178
Distance	230m	240m	250m	260m	270m	280m	290m	300m
Transmission Reliability	0.153	0.130	0.107	0.086	0.066	0.047	0.029	0.012

Link Estimation: We use EWMA (Exponential Weighted Moving Average) to estimate the link condition with $\alpha = 0.8$, $w = 1$. To simplify the calculation, we calculate the transmission reliabilities based on the received packet in the last 10 seconds.
Network Topologies: Firstly, for simple evaluation of GPR and SOAR work in a multiple flows wireless network, we simulate GPR and SOAR on the 3×3 and 5×5 grid topologies with two flows. Then we run simulations on various random topologies to verify that GPR works well as the network scale increases.

To compare the performance of GPR and SOAR, we choose the following metrics:

- **Throughput or Packet Delivery Ratio:** This is defined as the ratio of the number of packets received by the destination, to the number of packets originated by the source.
- **Delay:** The average duration takes from a packet is initially sent by the source to it is successfully received at the destination for the first time.
- **Path Length:** Path length is defined as the number of hops a packet takes to reach its destination.

5.2 Results and Evaluations

In grid topologies, we generated 2 CBR flows send data at the rate of 6Mbps, using the diagonal nodes as source-destination pairs. The smallest distance between any two neighbored nodes is 70m. The elementary results are showed by Table.2 and Table.3.

From these two tables, we see that the throughput decrease in wireless network as the path length grows. In the 3×3 grid topology, GPR takes advantage of the source-destination pairs are within the largest transmission range of each other. GPR increases the throughput by 22.41%, decreases the delay by 86.17% compare to SOAR. In the 5×5 grid topology, we see the throughput decreases with path length grows. Table.3 shows that GPR's throughput decrease slower than SOAR's, but the delay increases obviously. This is because some of the packet's holding and waiting time in relay nodes increases significantly than the others.Table.3 shows the throughput of GPR increases about 211% than that of SOAR.

Table 2. The elementary results for a 3×3 grid topology with 2 flows

Routing Protocol	Throughput	Average Delay	Path Length
GPR	95.44%	0.013	2.050
SOAR	78.04%	0.094	2.267

Table 3. The elementary results for a 5×5 grid topology with 2 flows

Routing Protocol	Throughput	Average Delay	Path Length
GPR	46.67%	1.173	4.135
SOAR	15.02%	0.437	5.023

Next, we run simulations on networks with 20-60 nodes placed randomly in different square areas. In each scene we selected 2, 3, 5 pair of nodes at random and generate CBR traffic with 6 Mbps data rate.

Fig.4 shows the throughput in different network scales with different number of flows. We see that the throughput decreases as the network scale increases, but GPR's decreases slowly than SOAR's. When 20 nodes uniformly distributed on a 250m×250m square, GPR can output SOAR from 54.08% to 312.07% as the network load increases. When the scale of network reaches 60 nodes in area of 450m×450m square, SOAR can not work at all while GPR still works properly.

Fig.5 plots the average delay of GPR and SOAR in different network scales and varied flow conditions. As the network scale reaches 60 nodes, SOAR can not work properly and Fig.5 shows this as delay reaches the maximum. In all kinds of network conditions, GPR outputs SOAR from 6.5% to as much as 35 times.

Fig. 6 shows the average path length the packets come through in diversity network conditions. It also shows SOAR does not work at all using the maximum value when the network scale reaches 60 nodes. We see that GPR decreases the average path length compared to SOAR. This verifies that GPR makes every transmission as long as possible to get the largest distance a packet can advance.

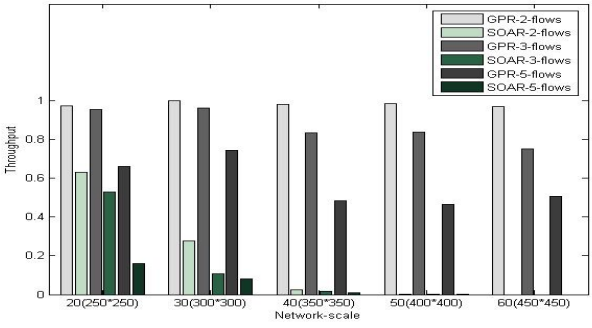


Fig. 4. The throughput of GPR and SOAR with different flow numbers in different network scales

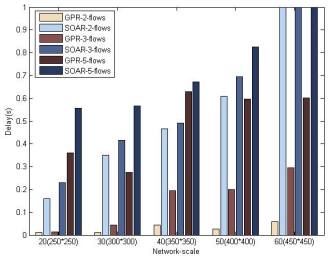


Fig. 5. The average delay of GPR and SOAR with different flow numbers in different network scales

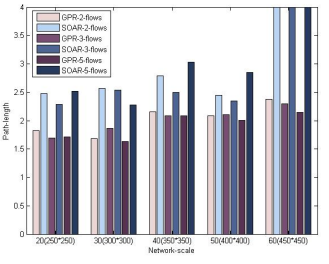


Fig. 6. The average path length of GPR and SOAR with different flow numbers in different network scales

6 Conclusion

In this paper, we present a novel opportunistic routing protocol, called GPR. It uses the geographic information of source-destination pair to get candidate subset. The relay nodes listen to the ACK messages from other candidates before their turn to send ACK and transmit the packet. If there was no ACK messages have been received, the nodes acknowledge the packet and relay it at the transmission-probability. In this way, the duplicate transmissions decreases with little coordinate overhead. GPR cooperates with the mac layer to prevent duplicate transmission or packet loss because nodes make the transmission decision incorrectly. We use simulations prove that the throughput of GPR can increase over 300% than reference method. The simulations also verify that GPR is scalable as the network scale increases.

Acknowledgement

This research was supported by the National Basic Research Program of China (973 Program) under grant No. 2006CB303000, National Natural Science Foundation of China under Grant No. 60473092 and No. 90612018.

References

[1] Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: A survey. Computer Networks Journal (2005)
[2] Chachulski, S., Jennings, M., Katti, S., Katabi, D.: Trading Structure for Randomness in Wireless Opportunistic Routing. In: Proc. of ACM SIGCOMM, ACM Press, New York (August 2007)
[3] Rozner, E., Seshadri, J., Mehta, Y., Qiu, L.: Simple Opportunistic Routing Protocol for Wireless Mesh Networks. In: Proc of the 2nd IEEE Workshop on WiMesh, IEEE Computer Society Press, Los Alamitos (2006)
[4] Biswas, S., Morris, R.: ExOR: Opportunistic multi-hop routing for wireless networks. In: Proc. of ACM SIGCOMM, ACM Press, New York (August 2005)
[5] Biswas, S., Morris, R.: Opportunistic routing in multi hop wireless networks. In: ACM HotNets (August 2003)
[6] Couto, D.D., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. In: Proc. of ACM MOBICOM, ACM Press, New York (September 2003)
[7] Draves, R., Padhye, J., Zill, B.: Comparison of routing metrics for multi-hop wireless networks. In: Proc. of ACM SIGCOMM, ACM Press, New York (August 2004)
[8] Jain, S., Das, S.: Exploiting path diversity in the link layer in wireless ad hoc networks. In: Proc. of the 6th IEEE WoWMoM Symposium, IEEE Computer Society Press, Los Alamitos (June 2005)
[9] Johnson, D.B., Maltz, D.A., Broch, J.: DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. In: Ad Hoc Networking (2001)
[10] The network simulator – ns-2, http://www.isi.edu/nsnam/ns/
[11] Perkins, C.E., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proc. of ACM SIGCOMM, ACM Press, New York (1994)
[12] Perkins, C.E., Royer, E.M.: Ad hoc on-demand distance vector routing. In: Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, IEEE Computer Society Press, Los Alamitos (February 1999)
[13] Mauve, M., Widmer, J., Hartenstein, H.: A Survey on Position Based Routing in Mobile Ad-hoc Networks. IEEE Network Magazine 15(6), 30–39 (2001)
[14] Basagni, S., et al.: A Distance Routing Effect Algorithm for Mobility (Dream). In: Proc. 4th Annual ACM/IEEE Int. Conf. Mobile Computing and Networking, MOBICOM, IEEE Computer Society Press, Los Alamitos (1998)
[15] Kranakis, E., Singh, H., Urrutia, J.: Compass Routing on Geometric Networks. In: Proc. 11th Canadian Conf. Comp. Geo., Vancouver (1999)

Towards a New Methodology for Estimating Available Bandwidth on Network Paths

Shaohe Lv, Xiaodong Wang, Xingming Zhou, and Jianping Yin

National Laboratory for Parallel and Distributed Processing National University of Defense Technology, Changsha Hunan, 410073, China
chi.shaohe@gmail.com, xdwang@nudt.edu.cn, xmzhou@nudt.edu.cn, jpyin@nudt.edu.cn

Abstract. This paper presents a novel methodology, called COPP, to estimate available bandwidth over a given network path. COPP deploys a particular probe scheme, namely chirp of packet pairs, which is composed of several packet pairs with decremental inter-packet spacing. After each pair chirp is received, COPP discovers all turning points, e.g. such packet pair that is interfered more seriously in contrast to its adjacent pairs, and give each point a distinct weight according to the degree to which the pair and its consecutive neighbors are interfered by cross traffic to yield an estimate. The final estimate is the average of the results of all chirps in a measurement episode. Two decision rules are developed to determine whether a packet pair is turning point. We test COPP in various simulations and find that COPP can provide accurate results with relatively less overhead while adapt to network variations rapidly.

1 Introduction

Estimating the bandwidth of an Internet path has become a hot issue recently. Knowledge of the path bandwidth can be put to good use in various scenarios, such as congestion control, server selection and network security and so on. It is observed that bandwidth monitoring and estimation have become increasingly important.

There have two metrics are commonly associated with bandwidth estimation — capacity and available bandwidth (avail-bw). The capacity of a link is the maximum transmission rate that link can transfer packets in the absence of competing traffic. In contrast, the avail-bw of a link at a time instance is defined as the maximum transmission rate that link can provide without interfering the existing traffic.

Consider a path, e.g. a sequences of links that forward packets from sender to receiver. The link with the minimum capacity determines the capacity of the path, while the link with the minimum available bandwidth limits the avail-bw, which we refer to as narrow and tight link respectively. In a time scale, the avail-bw of the path is the average of all the avail-bw at every time instance within the underlying scale. In this paper, we only focus on the measurement of path avail-bw in an interval.

Packet pair technique [1, 3, 5] has been used as one of the primary procedures to measure the capacity of a path. Recently, there has much work [10, 13], including the techniques we will propose, striving to estimate available bandwidth based on packet pair. When we focus on capacity, one of the most crucial issues is how to eliminate

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 487–496, 2007.

the interference caused by co-existing cross traffic. But for available bandwidth, what we mainly concern is that whether packet pair properly reflects the network dynamics, i.e. capturing the avail-bw and the burst of cross traffic. Aiming for this, we deploy a particular probe scheme named chirp of packet pairs which shows promising performance in simulation experiments.

Note that the time-varying nature of avail-bw requires that the measurement must be executed quickly in order to capture or adapt the network variation. In addition, as a reflection of network status, the actual avail-bw may be different from the measured one if the overhead is too high to change the network. Therefore, except the accuracy, we also concern the time and overhead in generating a final estimate of measurement tools. By exploiting the information in probes more efficiently, COPP that we introduce yields better tradeoff between good accuracy and low overhead.

The rest of this paper is organized as follows. In section 2, we briefly review the related work in estimating path avail-bw. In section 3, the description of COPP will be presented. We discuss the analysis of packet-pair that served as the fundament of COPP in section 4. Then we discuss the weighting process in section 5. And the simulation results are presented in section 6. Finally, we conclude in section 7.

2 Related Work

We briefly survey the many tools and techniques that have been proposed for estimating avail-bw of network paths. As for capacity estimation, we refer interesting readers to [11], which we omitted here due to space limitations.

Early techniques such as Cprobe measured the asymptotic dispersion rate rather than avail-bw. Many of the recently presented techniques fall into two categories: packet rate model (PRM) and packet gap model (PGM). There are several tools [10], actually, are mixture of PGM and PRM. PGM-based tools [13], in general, send pairs of equal-sized probe packets, and the increasing intra-pair spacing is used to estimate the volume of cross traffic, which is then subtracted from the capacity to yield the avail-bw. Tools such as [2, 6], also based on PGM, which is different for these tools send several trains and the one-way delay (OWD) of every probe packet is used to estimate the utilized bandwidth instead of the pair spacing. PGM assumes that the tight link is also the narrow link, and is susceptible to queue latencies at links except the tight link. Additionally, the errors introduced in capacity estimation would be propagated if the assumption in PGM that the capacity is known is violated. We pick Spruce [13] as the representative PGM-based tool for our experiments.

PRM-based tools [9, 12], are based on the observation that probe traffic sent at a lower rate than the avail-bw would be received at the sending rate (on average) — means that the probes would experience little interference. In contrast, if the sending rate exceeds the avail-bw, the received rate would be less than the sending rate and the probe packets tend to queue behind previous probe or cross packets—means that the probe packets are interfered more seriously. Thus, avail-bw can be measured by searching the turning point at which the interference varied remarkably. The increasing trend of OWD is used to determine this interference in Pathload [12], which is picked as the representative PRM-based tool for our experiments. The technique we present is also PRM-based but uses a pair as a minimum process unit rather than a single packet or train used in previous tools.

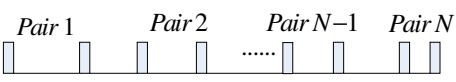


Fig. 1. Chirp of packet pairs: A Illustration

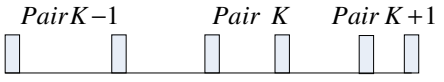


Fig. 2. The range to analyze the degree of interference of pair K when it is a turning point

3 COPP Methodology

In this section, we describe the basic idea and the measurement process of COPP. Some key issues such as the decision rules to determine whether a packet pair is turning point are discussed in next section.

We use measurement episode to indicate the time interval when a final avail-bw is reported by COPP. In each episode, several chirps of packet pairs are issued. For our purpose, a pair chirp as shown in Fig.1 preserves the follow property: the intra-pair spacing is gradually reduced to increase the sending rate of corresponding pair continuously, from the lower estimate bound to the upper bound, whereas the inter-pair spacing is kept large enough to avoid the interference between different pairs. The name COPP is, originally, from the probe scheme, e.g. Chirp Of Packet Pairs.

We use the term *turning point* to denote such packet pair that has been interfered more seriously compared to its previous neighbor. Associated with every turning point, there has a *turning bandwidth* (TB) that is the sending rate of the previous pair of the underlying turning point. Exploiting the decision rules depicted in section 4, COPP can discover all turning points and the corresponding TB. After each chirp is sent and traversed the measured path, COPP can obtain a temp partial estimate.

From the basic PRM observation, we know that turning point appears when the relationship between the sending rate of two consecutive pairs and avail-bw is altered. However, the burst nature of cross traffic may also trigger some "new" turning point. Thus, we should distinguish different "kind" of turning points and dispose them differently. More details are discussed in section 5. We now only present the solution of COPP: giving each point a distinct weight according to the degree to which the underlying point and its neighbors are disturbed. The follow equation 1 exhibits the derivation of the partial estimate for a chirp denoted by A, where $TB_1, TB_2, ...TB_k$ and $w_1, w_2 ... w_k$ denote all turning bandwidth and corresponding weight respectively in the chirp.

$$A = \sum_{i=1}^{k} (TB_i * w_i) / \sum_{i=1}^{k} w_i \tag{1}$$

Equation 2 shows the calculation of the final result in a measurement episode denoted by *Avail-BW*, where n is the amount of chirps launched in the episode and A_k is the partial estimate of the k-th chirp.

$$Avail\_BW = \sum_{k=1}^{n} A_k / n \tag{2}$$

There have several parameters to be set such as the packet size, the estimate bound, the amount of pair in a probe, the amount of pair in an episode, the function to

generate weight and the threshold used in decision rules, and so forth. Limited by space, we only discuss the analysis and selection of the threshold in section 5.3. The optimal settings depend on the underlying networks. The aim of the discussion in section 5.3, therefore, is to show the method or principle in parameter selection rather than to present a universal choice. We are in the process to implement the automatic setting of parameters according to the different network conditions.

4 Analysis of Packet-Pair

Packet-pair is at the heart of many bandwidth estimation techniques. We first ana-lyze the relationship of the one-way delays of two arbitrary packets traversed the same path, and further between the inter-packet spacing of a packet pair when started to be sent and when received. Then the decision rules to determine whether a pair is turning point can be derived.

4.1 Packet One-Way Delay Visiting

Since a network path is composed of several store-and-forward links, we first discuss the one-way delay of packet experienced in the single-link case. The total time spend in transmission through a link, L_k, consists of transmission latency, queue latency and propagation latency. The last metric is the time consumed in propagating signals through the link at the velocity of light and is free from different packet, which suggests $p_k^m = p_k^n$ where p_k^i denotes the propagation latency of the *i*th packet at the link. The middle metric, denoted by q_k^m, is the queue time of the *m*th packet from arriving at the link to starting to be processed. The first metric is the time between when the first bit of the packet starts to be transmitted and when the last bit already departs from the link, which can be represented as L/C_k where *L* is the packet size and C_k is the capacity of link L_k.

As for a path, Equation 3 pictures the OWD of the *m*-th packet with size *L* after traversed the path consisting of *n* links.

$$OWD_m = \sum_{k=1}^{n} (\frac{L}{C_k} + q_k^m + p_k^m) \tag{3}$$

We now focus on the relationship of the one-way delays of two arbitrary packets with the same size and transmitted across the same path of *N* physical links, i.e. the *m*th and *n*th packet. Note that the propagation latency of the two packets at each link is identical, the difference of the two one-way delays can be denoted by equation 4.

$$OWD_m - OWD_n = \sum_{k=1}^{N} (q_k^m - q_k^n) = \sum_{k=1}^{N} q_k^m - \sum_{k=1}^{N} q_k^n \tag{4}$$

4.2 The Packet-Pair Spacing Analysis

Consider a path of link $L_1, L_2 ... L_n$ with capacity $C_1, C_2 ... C_n$ respectively. We use t_m^k to denote the arrival time of the *k*th packet at link L_m. Specially, t_0^k marks the point

of the *k*th packet starts to be sent. Hence, for a packet pair (*P*1, *P*2) with size *L*, the intra-pair spacing at sender is denoted as $Spacing_s = t_0^2 - t_0^1$ and the spacing at receiver of the measured path as $Spacing_r = t_n^2 - t_n^1$. In addition, the one-way delay of the *k*th packet in the pair is represented by $OWD_k = t_n^k - t_0^k$. Then the spacing variation, *SV*, between at sender and receiver can be denoted by follow:

$$Spacing_r - Spacing_s = (t_n^2 - t_n^1) - (t_0^2 - t_0^1) = (t_n^2 - t_0^2) - (t_n^1 - t_0^1) \quad (5)$$

Then we obtain the equation (6):

$$SV = Spacing_r - Spacing_s = OWD_2 - OWD_1 \quad (6)$$

From Equation 4, one can observe that the one-way delay variation of different packets is determined by the difference of their total queue latency. According to Equation 5 and 6, the intra-pair spacing variation between at sender and receiver of a path is also identified by the underlying different queue latency. We know that the queue latency figures the interference from other traffic. The one-way delay variation of different packets and the intra-pair spacing variation, therefore, characterize the difference of interference experienced by the corresponding packets. Based on above investigation, the rules to determine whether a packet or pair is interfered seriously by cross traffic and whether a pair is turning point can be derived.

4.3 Decision Rules

Since there has no interaction of different probe pairs, the first packet of pair fairly reflects the state of the network consists of cross traffic only while the second packet captures the interaction of probe and competing traffic. In the simulation network described in section 6.1, we conduct a number of probe pairs, e.g. 1000 pairs, at the rate equal to avail-bw (4.6Mbps). We also select the cross packet just before the first packet of each pair as comparison. Fig.3 shows the cumulative distribution of the OWD of the probe and cross packets where *FP* and *SP* and *CP* refer to the case of first and second probe and cross packets respectively. We can see that the curve of *FP* and *CP* is close and differ from *SP* remarkably, which confirms the above states and one can further conjecture that the network status can be inferred from the information carried in all first packets.

Defining the threshold as $\varepsilon = (MD - AD)/k$ where *MD* and *AD* denotes the maximum and average one-way delay of all the first packets respectively and *k* is an empirical value. We believe that *MD* and *AD* represent the general experience of packet traversed the network with cross traffic only. The effect on estimate of different *k* will be inspected in section 5.

With the help of threshold, we first present the rule to determine whether a packet is distorted seriously: the *j*th packet is distorted seriously only if the underlying one-way delay satisfies $OWD_j > AD + \varepsilon$. Further, we define that a packet pair is interfered seriously by cross traffic if and only if the second packet of the pair is dis-torted seriously.

We now present the crucial rules to determine whether a packet pair is turning point. The first decision rule is as follows: The *j*th pair is turning point if it is interfered seriously and the previous one (the (*j*-1)th pair) is not interfered seriously.

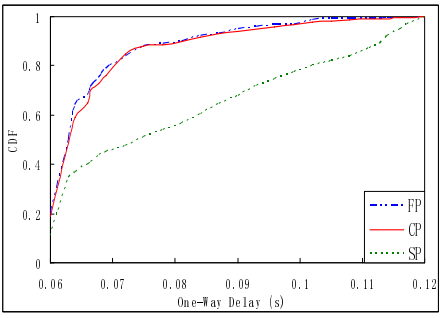


Fig. 3. The cumulative distribution of the one-way delay of probe and cross packets

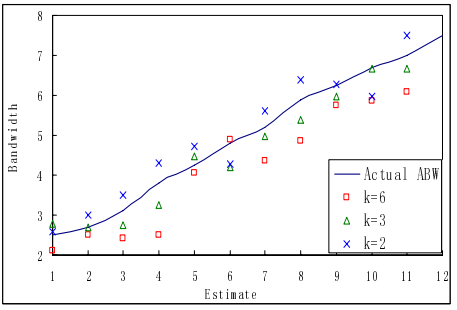


Fig. 4. Distinct Estimates of Different value of k, where x-axis represents different estimates and y-axis represents the bandwidth in Mbps

The second rule attempts to define whether a pair is turning point when the pair is not interfered seriously: if the variation of the intra-pair spacing of the jth and $(j-1)$th pair at sender and receiver satisfies $IV_j - IV_{j-1} \geq \varepsilon$ while the $(j-1)$th pair is also not interfered seriously, then the jth pair is turning point.

One should be noted here is that COPP need the cooperation of the two side of the measured path result from the use of one-way delay. Sender is in charge of sending probe traffic and calculating bandwidth, whereas receiver should send a special feedback packet indicates that the time instances when a probe packet is received. Thus, all timestamps at the emission and arrival of all probes are available to sender, from which the one-way delay and intra-pair spacing can be derived. In addition, since one-way delay is only used in comparison, there is no need to synchronize the time clock between sender and receiver.

5 Weighting Process

In this section we present the process giving each turning point a distinct weight. We first visit the difference of different kinds of turning points categorized by the corresponding main causality. Then the details of the weighting process are discussed. Finally we investigate the effect of the value k in the definition of threshold.

5.1 Turning Point Visiting

Available bandwidth characterizes the interaction between the ongoing and upcoming traffic. Several research show that Internet traffic exhibits LRD or self-similar property [7], which means that burst is a common property that must be addressed when we concern avail-bw on Internet path. The burst busy period when the network is busy in handling the cross traffic would reduce the avail-bw while the opposite burst idle period means that no cross traffic exists would increase the avail-bw. However, the effect of short-term burst on available bandwidth in a long-term time interval is limited and not deterministic.

Thus, it is of great importance to distinguish the turning point that is mainly due to burst from that mainly due to that the relation between probe sending rate and avail-bw is altered in the expected case. Further, in the context of the burst traffic, there have three kinds of turning points:

- Expected case: pair *P* is sent at rate higher than avail-bw while its previous neighbor is sent at rate lower than avail-bw, then the interference experienced by the two pairs would be obvious different, and the pair *P* is interfered more seriously and just the turning point.
- Busy period: when sent at rate below avail-bw, but encountering the burst busy period, probe pair *P* would be distorted more seriously than its former one, which suggests that the pair *P* should be identified as turning point.
- Idle period: when sent at rate exceeds avail-bw, but encountering the burst idle period, probe pair would experience little interference. Then the following pair *P* sent at higher rate would be interfered more seriously compared to this pair, which means that pair *P* is just the turning point.

Note that no interference occurs between consecutive pairs. So for those packet pairs around turning point that is mainly due to busy period, there has little interference experienced because their sending rate is still lower than avail-bw. As for those turning points that are mainly originated from idle period, both preceding and following pairs are interfered seriously for their sending rates exceed avail-bw. As for the expected case, the preceding pair would be distorted lightly since its sending rate is below avail-bw while the following one could experience severe interference since its sending rate exceeds the avail-bw. This means that the different kinds of turning point can be distinguished by the interference experienced by the point and its prior and follow neighbors.

5.2 The Weighting Process

We pick the turning point and the pairs that precede and follow the point as the range used to determine the degree to which the interference experienced by the packets within the range as shown in Fig.2 where pair *P* is turning point. The degree denoted by ρ can be calculated as $\rho = m / N$ where *m* and *N* denotes the amount of packets that are distorted seriously and the total amount of packets in the range respectively. In addition, in our experiments, *N* is 6.

According to the analysis of the prior section 5.1, the three kinds of turning points can be ordered and distinguished by the degree from little to large as busy period, expected case and idle period. Thus, functions taking the degree of each turning point as input can be used to give distinct weights to different kinds of points. In order to avoid that the turning points except in the expected case determine the estimate of avail-bw, the weight associated with these points must be relatively little. Finally, the weight function should be satisfied the follow property: the domain is (0, 1] and the maximum output is associated with when the input is 0.5; the output is gradually reduced until closed to 0 but always exceeds 0 when the difference of the input and 0.5 becomes increasingly large. Equation 7 show an example of the weight function, which is used in the follow simulation experiments. The function $\lfloor y \rfloor$ is a downhill

truncation function, for example $\lfloor 10.9 \rfloor = 10$ and $\lfloor 10.1 \rfloor = 10$. Note that the equation 7 is an empirical selection after a number of careful experiments and we do not claim that this function can be applied in all situations. Actually, the performance of the same function in distinct network is different and the general choice is too hard to find. Fortunately, the performance difference is not very distinct [19].

$$f(x) = \begin{cases} \lfloor 10x \rfloor / 10 & x < 0.45 \\ x & x \in [0.45, 0.5] \\ 1 - x & x \in (0.5, 0.55] \\ \lfloor 10(1-x) \rfloor / 10 & x > 0.55 \end{cases} \tag{7}$$

5.3 Threshold Discussions

Out of question, threshold defined in section 4.3 plays a crucial role in the execution of decision rules and weighting process. We examine the effect of different threshold by varying the value *k* in its definition. Fig.4 shows the different estimate with variable threshold under the network setup described in section 6.1.Clearly, the results are relatively better than other cases when *k* is 3.

When *k* is too small, actually, the corresponding threshold would be too large. Then only little packet pairs sent at rate exceeds available bandwidth could become turning points. On the other hand, due to the large threshold, there have fewer probes around the turning point that satisfy the condition to be identified as the packets distorted seriously, which suggests that the corresponding ρ would be reduced. For those turning points due to idle period, of which the sending rate exceeds avail-bw and ρ is greater than 0.5, the underlying weight would increase since the reducing ρ will make their ρ more close to 0.5. However, the weights of the other two kinds of turning points of which the degree are less than 0.5 originally, are decreased for the degree is reduced to be far more away from 0.5. In a word, the final estimate would exceed the actual avail-bw due to the over-weight of the turning point caused by idle period and the under-weight of the other two kinds of turning point.

Similar analysis could be applied to the case of too large *k*, and the final estimate would be too small. The conclusion is that the case of *k* is 3 is preferred, which is our selection in section 6.

6 Experiments

In this section, we used ns-2 [8] to validate COPP mechanism described in the above section. The comparison among Pathload, Spruce and COPP shows that COPP can adapt to network variation and provide accurate estimates with fewer overheads.

6.1 Simulation Setup

Fig.5 shows the experimental topology, where the measured path from Sender to Sink consists of *n*+1 links. Cross traffic is transmitted over *n* connections from S_i to $R_i (i = 1,2,..n)$. Except the connection from S_n to R_n that has *n*-1 links overlapped with measured path, all the other connections have only one link overlapped with measured

path. The bottleneck link of the measured path is at the center of the path with capacity 10Mbps, and all the remaining links are with capacity 15Mbps.

All link of the measured path has the same amount of cross traffic that consists of several Pareto traffic with shape parameter as 1.6. In addition, The cross traffic over every link of measured path have the same traffic class distribution (95% TCP and 5% UDP) as well as the packet size distribution, e.g. there are three type packet sizes, 1500/700/40Bytes that consumes the 5%, 85% and 10% percentages of total amount of cross traffic respectively.

In simulations, a chirp in COPP consists of 25 packet pairs and the estimate scale is [400K, 10M] which means that the difference between the sending rates of adjacent pairs is 400 Kbps. Equation 7 shows the function generating the weight. Additionally, after four chirps are processed, COPP report the average result as the final estimate.

We pick Pathload and Spruce as the references to COPP in experiments. We use the setting of them same as their authors suggest in [12, 13]. As for Pathload, a stream is composed of 100 packets with size of 800Bytes and the amount of stream to yield an estimate is determined by the iterative algorithm in Pathload. To obtain an estimate, whereas, Spruce needs 100 pairs with size of 1500Bytes. We test these tools in several topology but only report the results based on the case of $n=5$ since all conclusions inferred are similar regardless of different simulation setup.

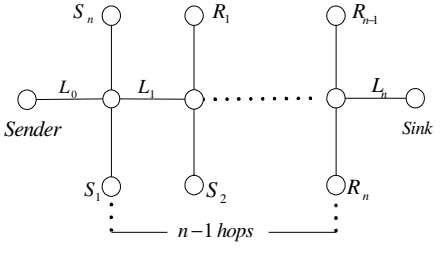


Fig. 5. Experiment Topology and the measuring path from Sender to Sink consist of $n+1$ links

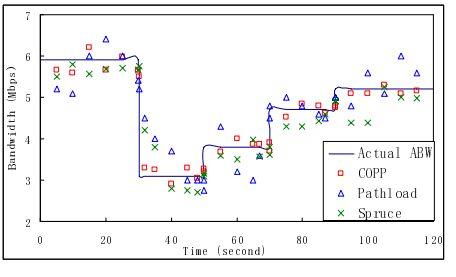


Fig. 6. Comparison among the estimate results of COPP, Pathload and Spruce

6.2 Experimental Results and Analysis

By controlling cross traffic, we make the avail-bw varied in time 30s, 50s, 70s and 90s whereas kept stable in the interval between two time points. Fig.6 shows the estimate results and actual avail-bw. Several conclusions can be drawn. First, almost all tools can provide approximately correct estimates, but their accuracy is distinct. For Pathload, the estimate can be changed remarkably when the actual bandwidth is invariant which suggests that the stability should be improved. As for Spruce which use probe of size 1500Bytes, the estimates are below the actual value commonly. Our technique, COPP, provides the best estimates in most case. Second, for agility, COPP is more agile when the variation of network is very acute, as for small variation, the performance of all techniques are comparable.

In almost 90% experiments, one can see that the estimate of COPP with three chirps in an episode is fairly accurate. The total amount of overheads is thus trivial for

network. As compared to other two techniques, the overhead of COPP is less than Pathload significantly and at most equal to Spruce. We also compare the intrusiveness of three tools in short-term interval. Note that the minimum process unit in Pathload is a sequence of 100 packets while in COPP and Spruce is a packet pair, and further all pairs in Spruce are sent back-to-back, we can conjecture that COPP is not more intrusive than the other two tools. In summary, COPP is an accurate and non-intrusive tool and can reflect the dynamic variation of network quickly in most cases.

7 Conclusions

This paper presents a novel methodology for estimating available bandwidth over a network path, called COPP. The relationship between sending rate and avail-bw is characterized through the experienced interference of probes inferred from the variation of pair dispersion and/or OWD. We describe the maim process of COPP and derive two decision rules to determine whether a packet pair is turning point from the analysis of packet-pair. We also investigate the settings of estimation parameters. We evaluate COPP by simulation and find that it can provide relatively accurate estimate with less overhead and adapt to network variation. The focus of our future work is to evaluating COPP in realistic network and integrating it with network applications.

References

1. Dovrolis, C., Ramanathan, P., Moore, D.: Packet Dispersion Techniques and A Capacity Estimation Methodology. IEEE/ACM Transactions on Networking (October (2004)
2. Min, L., Jinlin, S., Zhongcheng, L., Zhigang, K., Jian, M.: A New End-to-End Measurement Method For Estimating Available Bandwidth. In: proceedings of ISCC 2003 (2003)
3. Liu, X., Ravindran, K., Loguinov, D.: What Signals do Pakcet-Pair Dispersion Carry? In: proceedings of IEEE INFOCOM 2005 (2005)
4. Man, C., Hasegawa, G., Murata, M.: Available Bandwidth Measurement via TCP Connection. In: IFIP/IEEE MMNS, IEEE Computer Society Press, Los Alamitos (September 2004)
5. Jacoboson, V.: Congestion Avoidance and Control. In: Proc., SIGCOMM, USA (September 1988)
6. Lakshminarayanan, K., Padmanabhan, V., Padhye, J.: Bandwidth Estimation in Broadband Access Networks. In: IMC, Internet Measurement Conference (October 2004)
7. Karagiannis, T., Molle, M., Faloutsos, M., Broido, A., Nonstationary, A.: Poisson View of Internet Traffic. In: proceedings of IEEE INFOCOM (April 2004)
8. NS version 2. Network Simulator, Http://www.isi.edu/nsnam/ns
9. Kiwior, D., Kingston, J., Spratt, A.: Pathmon: a Methodology for Determining Available Bandwidth over an Unknown Network. Tech Report (2004)
10. Melander, B., Bjorkman, M., Gunningberg, P.: A New end-to-end Probing and Analysis Method for Estimating Bandwidth Bottlenecks. In: GLOBECOM (2000)
11. Prasad, R., Dovrolis, C., Moore, D.: Bandwidth Estimation: Metrics, Measurement Techniques and Tools. IEEE Computer Society Press, Los Alamitos (2003)
12. Jain, M., Dovrolis, C.: End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. In: proceedings of SIGCOMM (August 2002)
13. Strauss, J., Katabi, D., Kaashoek, F.: A Measurement Study of Available Bandwidth Estimation Tools. In: proceedings of ACM IMC (2003)

Design and Realization of Multi-protocol Communication Model for Network Security Management System

Shouling Dong and Jiaming Luo

School of Computer Science and Engineering,
South China University of Technology,
Guangzhou, 510640, P.R. China
{sldong,sejmluo}@scut.edu.cn

Abstract. The purpose of this paper is to set up an efficient and secure data transmission channel to transfer the security data between the management platform and the distributed security agents on the internet. We analyses the basic principle and characteristic of the data security transfer and multi-protocol communication model in the distributed management network base on new network management and security data transfer technology, and proposed the actual system structure of this model. Furthermore, this model carries on a lot instructive test and get well along with opening interface, real-time management, expansibility, human interface and crossing flat ability, which guarantee the successful application of a distributed network management system. Moreover, the system applies itself successfully to campus network testing.

1 Introduction

In today's network, various stand-alone security servers and/or proxies are used to provide some sort of piecemeal security, such as an authentication server or an authorization and access controller [1]. However, these servers only resolve certain types of security problems. It makes the system administrators' work complicated and inefficient because multiple equipments should be monitored concurrently in order for them to make a judgment on the integrated security status of the entire information assets. If the stand-alone security servers are worked separately, it would result in the waste of resources, also makes the lack of linkage between the systems and correlation analysis.

Recently, there has been a great interest in developing security systems that manage the entire security equipments [2]. With the prevalence of the Internet and network technology, people began to consider network management base on Web, a Web-based network management system is to allow the Web browser for network management.

Web-based network management model has the realization of two ways. One way is called the manager/agent fashion, which has a main management platform to control stand-alone security servers or devices. Multiple devices, known as agents, are distributed in the network environment, such as mail filtering system, IDS server,

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 497–507, 2007.

firewall etc. The management platform plays the role of a manager in this fashion to centralize the management of security devices so that the devices can work cooperatively. For example, comprehensive knowledge of the security status of the devices can easily be provided to the system administrator. The second way is to achieve embedded. Web function was embedded into device, each device has its own Web address, and administrators can directly access the device through browser, which is usually used for the small office network.

As concerned above, it is possible to use manager/agent fashion, creating a unified remote management center to solve the above problem.

In order to manage the distributed remote agent, it needs to address the following issues.

Different form traditional distributed network management systems, management center needs to cross firewall safety. There are multiple networks protocols and multiple routings, such as HTTPS, SMTP, FTP, SNMP, SSH and so on. It is more convenient to cross firewalls or monitoring procedures and arrived at the client by using multi protocols. Some devices may need to be accessed through different transport protocols. [12] For example, a mail filtering system may get e-mail samples that are transferred through SMTP. Deployment files may be transferred through FTP. We present the multi-protocol Communication technology. We can choose different protocols to communicate over a connection between client and server depending on their network environments, so that the data can traverse firewall to the distributed network management system.

Security issue is often considered in distributed systems too [10]. Different kinds of data are transferred between the management platform and the distributed security agents on the Internet, such as deployment files, logs, e-mails, notifications etc. It is necessary to create an efficient and secure data transmission channel to transfer the data. We discuss the secure data transmission channel to solve the problem.

The remainder of the paper is organized as follow. The next section gives the design of multi-protocol communication model; section 3 describes the realization of the model on mail filter system; Appraisal of the model is presented in section 4; finally, section 5 providing some concluding remarks.

2 Design

2.1 Design Goal

The security management platform must receive data for management form remote front-agent. The data will be analyzed, processed and saved in some way or arithmetic. Manager would modify the deployment and attributed base on the data, then some data may return to the agents.

There will be many data need to be transferred between security management platform and agents. The data may be various according to the types of the agents. Data need to be transferred between security management platform and an email filtering system agent, for example, may include the device's running state, server running state, agent's configure information, filter arithmetic parameter, samples, data logs and etc. These data can reduce to four types below:

(1) XML format

Such as system attributes, system parameters, configure information, user info, and system security events may in the format of IDMEF in XML Schema.

(2) Email format

Samples of mails, including spam mails, virus files and etc.

(3) Data logs

System running log files in agents and security management platform.

(4) Alarm and notification arise from agents

In order to make transmission between agents and the management platform more effective, it's necessary to choose different network protocols and security levels to transmit data according to some communications policies. Thus, we propose multi-protocol communication model for agent and management platform.

The design goal of the system is to use common system structure to provide a common, flexible solution to the transmission of security data in network environment, thereby raising the level of distributed network management and interoperability and scalability of communication model in network management system. It mainly includes the following design objectives.

(1) Provide multi-protocol (including HTTP, RMI, FTP and SMTP) adaptation of the transmission channel, for the efficiency of transmission between front-end agents' equipments and back-end security management platform.

(2) With certification, authentication, access control, digital signature, data encryption and other security mechanisms and communications control in the transmission channel to provide data security assurances for end-to-end transmission.

(3) Implement definitions and interface of data transmission based on the XML standard, as well as the security transmission protocol standards.

(4) Management Console can use different transmission protocols and policies to all kinds of information, which can be configured automatically by the system or by user, User can also customize security level of transmission of information.

(5) Communication model is portable and scalable, and high universality. Through standard data interfaces, it can integrate with a variety of third party front-end data acquisition systems.

2.2 Characteristics of Multi-protocol Communication Model

Base on JMX. JMX is so scalability, low implementation cost, and compatibility that it is suitable for developing a distributed network management system [3].

The design bases on JMX three level management framework and network data secure tunneling technology [4], [5]. The model consists of communication server, secure transmission tunneling, and communication client.

The server component is in JMX distributed service level, which provide distributed management interface accessible from remote management application. Secure data transfer tunnel constructs a generic and secure transmission model for management using the SSL (Secure Socket Layer) [6]. The client component manages and monitors resources implemented by standard or dynamic MBeans.

Figure 1 illustrates the structure of the distributed network management system.

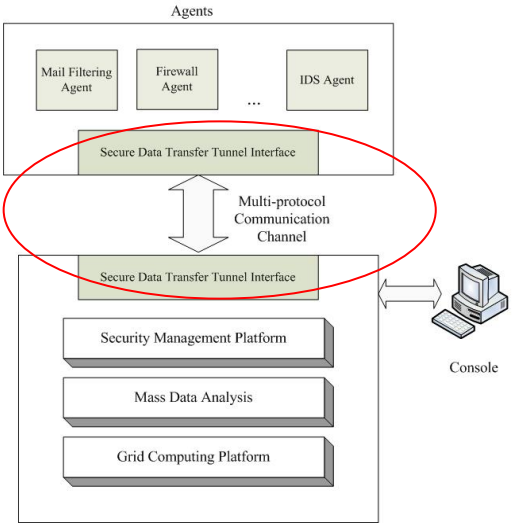


Fig. 1. Structure of the distributed network management system

Generic Interface of Protocol Adapter. The goal of our model is to define generic and unified interfaces. By implementing the interface we can use a connector base on a protocol that is not defined in the JMX Remote API standard. A typical example of this is a connector based on a protocol that uses HTTP/S. Other protocols are also possible.

This generic interface makes it possible to use various protocols to communicate with client and server dynamically.

See section 2.4 for more information.

Data Secure Transmission. The data need to be transferred over a secure channel, which means the data should be encrypted, authenticated etc.

We take account of the security mechanism of the system so that the connection between client and server uses the Secure Socket Layer (SSL). See section 2.4 for more information about the secure transmission structure.

2.3 Framework

According to the ideas above, multi-protocol secure transmission mechanism is designed and developed base on JMX framework.

In JMX framework, the whole design is simplified since JMX provide an integrative network management application development environment, and a useful, flexible, and dynamic framework. It also provides a series of JAVA classes and tools, which make developing dynamic and extensible network management software simpler.

Figure 2 shows our design of the multi-protocol communication model base on JMX.

Let's pay attention to the distributed service level of the model of which is the most important part. The interfaces and components in distributed service level

provide connectors through which administrator, agent and MBean can interact with each other. MBean is exposed by agent level, and massages are transmitted by protocol like HTML or SNMP through protocol adapter. This level issues the manage information from management platform to many agent service components. It integrates manage information from various agent service components into logical views, communicates with users and provides safeguard.

Here connectors and protocol adapters provide interface for remote manager to access agent through different protocol or remote method calls.

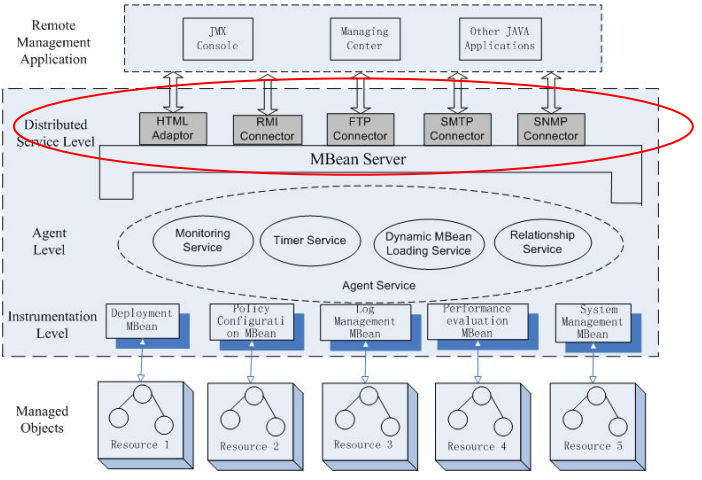


Fig. 2. The multi-protocol communication model

JMX agents can have any adapters and connectors, so secure management platform can access agent in multiple way.

JMX provides API to access Mbean resources, and it can be used to define user's protocol adapter. User-defined protocol adapter can quickly add to the agents that are already in use, so it can ensure the dynamic activities. It is the sticking point of the network management.

For our project, we use HTTP, FTP, RMI, and SMTP protocol.

2.4 Secure Data Transmission Channel

Secure Data Transfer channel module is one part of multi-protocol communication model, in the design of the channel is to meet the requirements for data transmission of a variety of network management applications, that is, data communications confidentiality, integrity, data source authentication, and prevent malicious attacks; In addition, the transmission channel for security management application provides a convenient interface, so that the channel can be very simple integration to other network management system in China.

This secure transmission channel includes certification, data source authentication, message integrity assurance, data encryption, data decryption and transmission control. Figure 3 illustrates this structure.

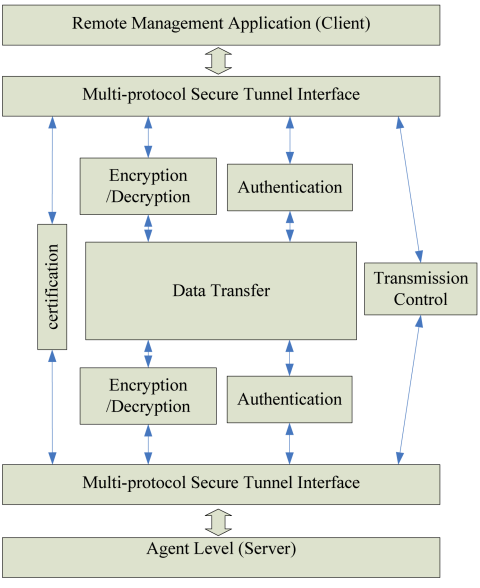


Fig. 3. Logical structure of secure transfer tunnel

Now we discuss each part of the structure.

Protocols. In the network environment, there are a lot of data transmission protocols, such as file transfer agreement (FTP), Simple Mail Transfer Protocol (SMTP), the HTTP protocol. These have their different characteristics in transmission speed, data encryption algorithm, transmission security, data format support, and other areas, by choosing a different protocol can bring different aspects of facilities.

It is very common to use FTP for file transfer, this approach has the advantage of widespread applicability, easy to use, users do not need additional preparation of the application software (usually integrated into the operating system or network protocol); the drawback is the lower transmission reliability when transferring large files. Hence it is not suitable for the application that requires large files transmission and high precision.

With the development of Web technology, using the traditional approach is bound to affect the actual application and promotion of file transfer; WWW technology's application also requested a new file transfer protocol and methods, which lead to the use of HTTP for file transfer.

Using SMTP, the host of sender and the mail server directly connected, thereby establishing a channel from the sender to the receiver directly, avoiding network firewall.

With the Java Remote Method Invocation (RMI) form enables application called Remote Application Packaging of the square France.

This shows that the different methods or protocols suit different application needs. The multi-protocol communication model should consider the advantages of various

transmission protocols, choosing different protocols and communications policies [13] under various data formats, systems' situations, and consider expansion of the system. After the system development to support more data formats and additional transmission requirements, it also can guarantee the transmission between management platform and the agents to be safe, accurate and timely. The model provides secure communications for a variety of transmission protocol support, including HTTP, FTP, SMTP, and RMI.

Certification. The main function of the certification part is to complete the certification between client and server. In general, Certification could be one-way or two-way [8]. The secure transmission channel was only server-to-client authentication, which is completed in the first phase of the connection. This is an effective way to avoid the "MAN-In-The-Middle" attacks. To enhance system security, servers need to provide CA presented to it by the CA certificate, and the client verifies the CA certificate from the server, if the certificate errors, authentication failure.

Authentication and Data Integrality. The main function this part is to complete the data source authentication and message integrity checking. The technologies, including digital signature technology and symmetric information authentication code technology, that is, HASH technology, Key commonly used hash algorithm HMAC-SHA and HMAC-MD5.

Encryption/Decryption. The main function of Encryption/decryption part is to guarantee the confidentiality of data, sender and receiver use common encryption / decryption algorithm, the shared key, encrypt data after earlier identification, and the receiver decrypts data using the same data decryption algorithm as the sender's.

Transmission Control. Transmission control part is mainly responsible for the communication of data processing flow control. It is responsible for the storage of algorithm group, the shared key; responsible for the process of data in accordance with specific strategies and flows, responsible for handling errors and control flow, and so on.

In this system, the servers and the client of the communication model both have the transmission control procedures. When server-side components have started, they will regularly report their conditions of the system to client. Once the client doesn't connecting communication servers, it will send a warning message. In addition, client has error control processes to deal with data loss in the process of transmission or other abnormal situation, through several attempts to reconnect, retransmission and clean data, in order to ensure the restoration after the mistakes as much as possible.

2.5 Agent Server Components

Agent components provide registration of Mbean, communications between long-distance customer management applications and managed objects, and transmiting information etc.

3 Realization of the Model

We realize the model between management platform and agents of e-mail filtering system. The management platform adopts MVC design pattern, uses Servlet and JSP technology. It uses Tomcat server to run Servlets and JSP in Web applications. The Mbeans of the agent will be run with Jboss.

The data that can be managed include agents' running status, server's running status, deployment files, filtering algorithm parameter, mail samples, data logs, etc. These data reduce to several data formats: XML [7], e-mail, and log files.

3.1 Realization of the Multi-protocol Transmission Channel

RMI Connector. It contains three components: RMIConnectorClient, RMIConnector Server and RMIClientFactory. RMIClientFactory Establishes a connection to server by calling RMIConnectorClient, then the RMIConnectorServer will return a RMI client object. The management center can invoke local methods which defined in Mbeans.

SMTP Connector. SMTP protocol connector use Sun JavaMail component. Management center use SMTP to send the data from management center in the form of e-mail attachments to the accounts specified by management procedures. The agent receives mail and attachments through the POP3 protocol.

FTP Connector. The first step, establish FTP Connecter Server in agents. Second, establish FTP Connecter Client in management center. The third, when agent server started, register FTP Conneter MBean, and started listening service ports (such as 21). The forth, client sends a request to FTPConnector Server for a two-way data transfer.

HTTP Connector. HTML server (such as Tomcat 4.0) has achieved the SSL (Secure Socket Layer) [9], [11]. Therefore, client and server-side can be in HTTPS connection.

3.2 Protocol Adapter

Figure 4 shows the module chart of multi-protocol adapter, which is in agent server. The protocol server in agent level may startup first to detect the request from client.

ProtocolTest is used to test whether the protocol is usable and the time for transmission, and makes a result for protocol factory.

ProtocolFactory chooses one protocol to use according to the test result from protocol test module.

Commpolicy module has self-adapting transmitting policy and real-time network status logs that can help protocol factory to choose which protocol is the best to use.

ProtocolClient creates a connection with protocol server and transfers encrypted data. Both protocol client and server use unified interface. Users can add their own defined protocol or delete some of the protocols easily.

CommunicateMrg manages the whole transmission process and handle errors.

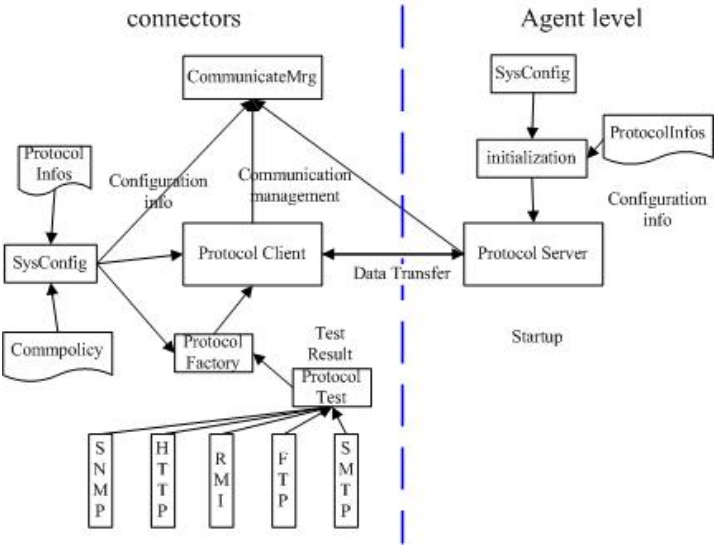


Fig. 4. Module chart of multi-protocol adapter

4 Appraisal of the Multi-protocol Model

We realize the model in our distributed network management system and make some instructive test. The test result shows that the multi-protocol mechanism gets along well with the system. Data can be transmitted by every protocol we defined. When one protocol is unable to use, the system can detect another protocol automatically. It supply friendly interface, figure 5 shows one configuration interface of the system. It allows user to configure parameters of protocols, such as the username and password of FTP.

Fig. 5. Deployment page of FTP

Fig. 6. Protocol testing

It has perfect security, using SSL to encrypt. It is transparent to users, while they can browse the websites as usual. It can be extensible because of generic interface of protocol adapters. When a new protocol is needed, the user can define their own protocol, with no changes on others.

We measured the time required for transmitting the data. It took less than 1 second to test the health of protocols and less than 2 seconds to send a general deployment file of 2K from management platform to the agent, so it gets along well with real-time management. Figure 6 shows the interface for testing protocol.

5 Conclusion

This paper discusses the model of secure data transfer between management platform and the distributed security agents using multi-protocol technology in network management. This communication mechanism bases on JMX framework, and fully considers the validity and security of data transfer. We realize the model. It provides good performance and expansibility under campus network testing. Base on our work, it is possible to establish data transmission strategy more intelligently, according to systems and networks' real-time status data, which would be our next work.

Acknowledgments for the Paper

Our colleagues and teachers have contributed greatly to the Research of the distributed network management system. Thanks all of you for your help and constructive criticisms. Many thanks to Pu Wang and Gongming Jiang.

References

1. Erfani, S.: Security Management System Functional Architecture for Enterprise Network. 0-7803-5864-3, IEEE, Honolulu, HI, 977-978 (2000)
2. Lim, Y., Kim, M., Jeong, A.: An Enterprise Security Management System as an ASP Solution. In: ICHIT 2006, 0-7695-2674-8/06, Cheju Island, Korea, pp. 548–555. IEEE Computer Society Press, Los Alamitos (2006) 0-7695-2674-8/06
3. JMX 1.2 Specification. Sun Microsystems,Inc. (April 2007), http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/
4. Sullins, B.G., Whipple, M.B.: JMX in Action. Manning Publications Company, Greenwich, Connecticut (2002)
5. Perry, S.J.: Java Management Extensions. O'Reilly, Sebastopol, USA (June 2002)
6. Freier, A.O., Karlton, P., Kocher, P.C.: The SSL Protocol Version 3.0 (November 1996), http://home.netscape.com/eng/ssl3/draft302.txt
7. Dong, S., Tan, Y., Zhang, L.: Extensible network security policy management system. Journal of Dalian University of Technology, China, 172-175 (2005)
8. Prem Kumar, G., Venkataram, P.: Security management architecture for access control to network resources. IEE Proc.-Comput. Digit. Tech. 144(6) (November 1997)
9. Berbecaru, D.: On Measuring SSL-based Secure Data Transfer with Handheld Devices. IEEE Computer Society Press, Los Alamitos (2005)
10. Cao, F., Dini, P.: Providing Secure End-to-End Session Tracking in Distributed Multi-Protocol Environments. IEEE Computer Society Press, Los Alamitos (2006)
11. Mraz, R.: Secure Blue: An Architecture for a Scalable, Reliable High Volume SSL Internet Server. IEEE Computer Society Press, Los Alamitos (2001)
12. Epstein, S.: Using Multi-Protocol Encapsulation Technology to Develop Optimum Data Broadcast Systems. 2000 The Institution of Electrical Engineers (2000)
13. de Albuquerque, J.P., Krumm, H., de Geus, P.L.: Policy Modeling and Refinement for Network Security Systems. IEEE Computer Society Press, Los Alamitos (2005)

Enhanced and Authenticated Deterministic Packet Marking for IP Traceback

Dan Peng, Zhicai Shi, Longming Tao, and Wu Ma

Information Engineering Institute, Dalian University, 116622, Dalian Liaoning, China
{pdlu10501003,szc1964,vilon888,mwps}@163.com

Abstract. The rising threat of cyber attacks, especially distributed denial-of-service (DDos), makes the IP traceback problem very relevant to today's Internet security. In this paper, a novel deterministic packet marking scheme called PN-DPM for IP traceback is presented. Through a unique technique: path numbering, our scheme provides ISPs a feasible solution to make IP traceback as a value-added network service, which allows the victim not only to detect and filter spoofed DDOS attacks immediately, but also to obtain more accurate information about the source of the attacks from the corresponding ISPs by the authenticated marks, even after the attacks has been completed. Our techniques feature low network and router overhead, low computational load for victim, and support incremental deployment. In contrast to previous work, our technique has significantly higher feasibility by considering much more aspects of IP traceback technology from practical perspective.

1 Introduction

Due to the deficiencies of TCP/IP architecture, cyber attacks always follow the development of network. Furthermore, IP header fields, including the 32-bit source IP header field can be readily modified. So the attackers can forge the source address of a packet, effectively hiding its true origin. Especially in recent years, Distributed Denial-of-Service (DDoS) attacks, one of the most effective attack ways, pose a major threat to the availability of Internet services. A DDoS attacker can greatly reduce the quality of a target Internet service or even can completely break the network connectivity of a server by persistently overloading critical network or system resources of the target, such as network bandwidth, router processing capability, or CPU/memory at the target machine. Moreover, to hide the sources of attack traffic and circumvent DDoS defense mechanisms relying on inspecting IP header fields, DDoS attack programs generally fill IP header fields, especially the 32-bit source IP address, with randomized values. To defend against spoofed attacks, especially DDOS, a technique called IP traceback to locate the actual source of attack packets is proposed and evolving rapidly in recent years.

While many IP traceback techniques have been proposed, they mostly have shortcomings that limit their usability in practice (we discuss more details on related work in section 2). One promising solution, referred as DPM, was first proposed in [1] and an enhanced scheme was shown in [2][3]. DPM is a deterministic packet marking algorithm which is scalable and simple to implement, and introduces no bandwidth and

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 508–517, 2007.

practically no processing overhead. In addition, attacks which are composed of just a few packets can be traced back by DPM. Unfortunately, as we will show in our analysis in section 3, this approach has an internal shortcoming when confronts some new network techniques, such as Network Address Translation (NAT). This approach is also vulnerable to compromised routers [4].

In this paper we present a new IP marking scheme called PN-DPM to solve the IP traceback problem. Besides all the advantages of DPM, through encoding the paths between the edge DPM routers of domain or subnet and their upstream routers, the victim can use the path Identifiers not only as path fingerprints to detect and filter DDoS traffic, but also as the efficient authentication of routers' marking. Furthermore, after reconstructing the edge marking and decoding the path identifiers, more accurate information can be utilized for post-mortem capability, which may be provided as a value-added service by ISPs.

2 Related Work

Many DDoS defense mechanisms have been proposed in recent years. These schemes can be roughly categorized into four classes: attacker-end based, network-based, victim-end based, and hybrid [5]. Firstly, owning to no direct benefit to the source ISPs themselves, Attacker-end based and network-based schemes may not be put into practice in the near future. Secondly, due to the lack of built-in security mechanisms in the current Internet infrastructure, the victim-end schemes may always be circumvented by attackers when the defense mechanisms are acknowledged. Lastly, hybrid schemes may be the best solution, in which ISPs provide some network support to the victim as value-added services, and victim can utilizes them to defend against or trace the attack traffic. Up to now, several hybrid schemes have been proposed, but they mostly have not considered ISPs' benefit for network support. In a scheme [5], each participating router marks a hash value in the Identification field of an IP packet, according to the router's IP address and the old hash value in the IP header. In this way, an IP packet will arrive at its destination along with a unique identifier representing the path it has traversed. Since the marking is deterministic, packets traversing the same path will share an identical path identifier. With this scheme, the path identifier of a packet will provide the victim the ability to identify spoofed IP packets. Since most of current DDoS attack tools generate spoofed IP packets, this schemes can effectively defend against spoofed DDOS. However, it did not have the post-mortem capability of locating the sources of attacks to charge the attackers, and deal with other types of DDOS attacks, such as Distributed Reflector DOS (DRDOS) which not mainly consisted of spoofed packets.

IP traceback technique was mainly developed to be as a DDOS countermeasure. But it focuses on identifying the origins of spoofed attacks, rather than stopping these attacks. Thus, it does not provide immediate help to victims. According to the processing mode, IP traceback can be classified into two categories: deterministic and stochastic [4]. And the Deterministic Packet Marking (DPM) and Probabilistic Packet Marking (PPM) are the representative schemes respectively. There have been many modifications and discussions on PPM since it appeared. Yet PPM has many drawbacks such as heavy computation load, high false positive, spoofed marking and

so on [4]. Furthermore, PPM was fully designed to locate the sources of DOS or DDOS attacks which consisted of a large number of attack packets. Besides low router and network overloading like PPM, the task of ingress address reconstruction in DPM as a simple and direct solution to IP traceback, is much simpler than the task of path reconstruction in PPM. As a result, DPM not only may handle large-scale DDoS attacks better, but also has the potential to tackle reflector-based DDoS attacks, and attacks which are composed of just a few packets. Moreover, a service provider can implement this scheme without revealing its internal network topology.

A DDOS-defending mechanism based on IP traceback can be classified as a hybrid scheme. The type of this defending scheme not only can provide immediate help to victims, but also can be used to identify the origins of spoofed attacks. An IP traceback method based on PPM is employed to construct the attack graph, and subsequently IP packets marked with one of network edges in the attack graph are discarded [6]. This scheme suffers from the large number of packets required to construct the attack graph. And, it may misclassify legitimate packets as attack packets if legitimate packets ever traversed the network edge in the attack graph. Up to now, no scheme based on DPM has been developed to defend against DDOS. Our scheme is inspired by the above works.

3 Enhanced and Authenticated Deterministic Packet Marking

Our proposed algorithm is essentially a packet marking algorithm. We first observe the drawbacks of Deterministic Packet Marking (DPM) which was proposed in [1][2][3], and then try to address them in our proposal.

3.1 Observations of DPM

In DPM, The 16-bit Packet ID field and the reserved 1-bit Flag in the IP header will be used to mark packets. Each packet is marked when it enters the network. This mark remains unchanged for as long as the packet traverses the network. The packet is marked by the interface closest to the source of the packet on the edge ingress router, as

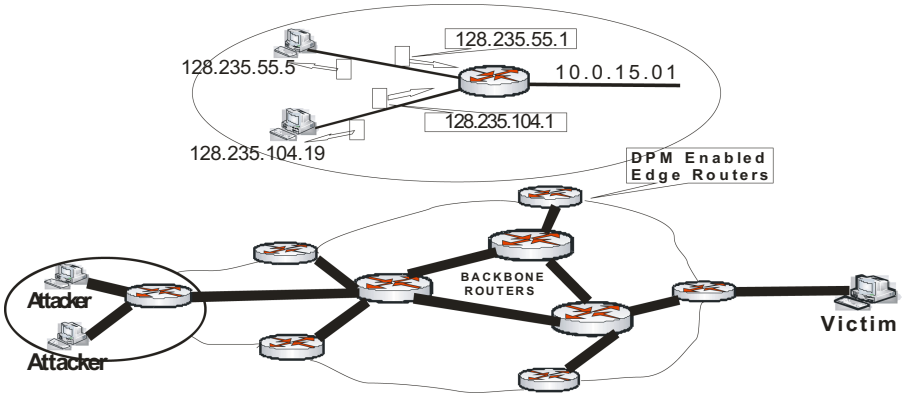


Fig. 1. Deterministic Packet Marking (DPM)

shown in Fig.1.The interface makes a distinction between incoming and outgoing packets. Incoming packets are marked; outgoing packets are not marked. This ensures that egress router will not overwrite the mark in a packet placed by an ingress router.

We make the following observation about the ingress-marking schemes in DPM. Now, the 32-bit address field of the current IP protocol limits the number of possible addresses, the tremendous growth of the Internet threatens to eventually make IP addresses a scarce resource. Many organizations' networks use private addresses in their own networks, and only have several public addresses to communicate with other organizations. Moreover, with the technique of Network Address Translation (NAT) appearing, the computers not only can use IP address at will, but also can access the public network, such as Internet. In this situation, the same ingress address marks may appear if the DPM enabled routers in DPM are placed in private networks, as shown in Fig.2, and DPM will have higher false positive. If the DPM enabled routers in DPM are only placed in public networks, the ingress mark may be only a address of a subnet or domain, and this provide less accurate information about source of attacking traffic. We observe that it is a result of using one-dimension address space in DPM, which can not reflect the hierarchy of current routing infrastructure. Furthermore, a fundamental shortcoming of previous DPM marking schemes is that the packet markings are not authenticated. If a router is compromised, it can forge markings and hence lead the reconstruction to wrong results. Even worse, the victim will not be able to tell if a router is compromised just from the information in the packets it receives. Finally, the marking field is only 17 bits. The small space for storing mark severely limits the capability of traceback in DPM.

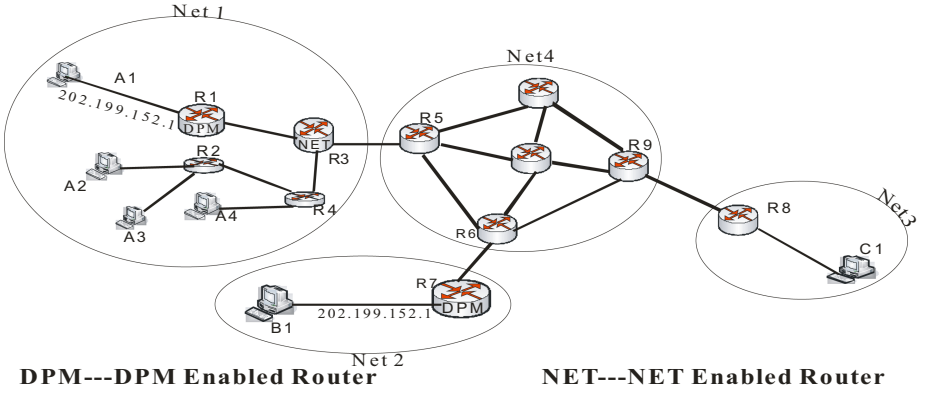


Fig. 2. A Scenario of The Same Address Marks Appearing in DPM

3.2 Overview of PN-DPM

To reflect the hierarchy of current routing infrastructure for gaining more accurate information of the sources of attack packets, a two-dimension address mark scheme is proposed. In our scheme, there are two types of the participating routers: DPM-enabled routers and PNM-enabled routers, as shown in Fig.3. DPM-enabled routers are deployed at the edge of a domain or subnet to mark each packet traversing them by the

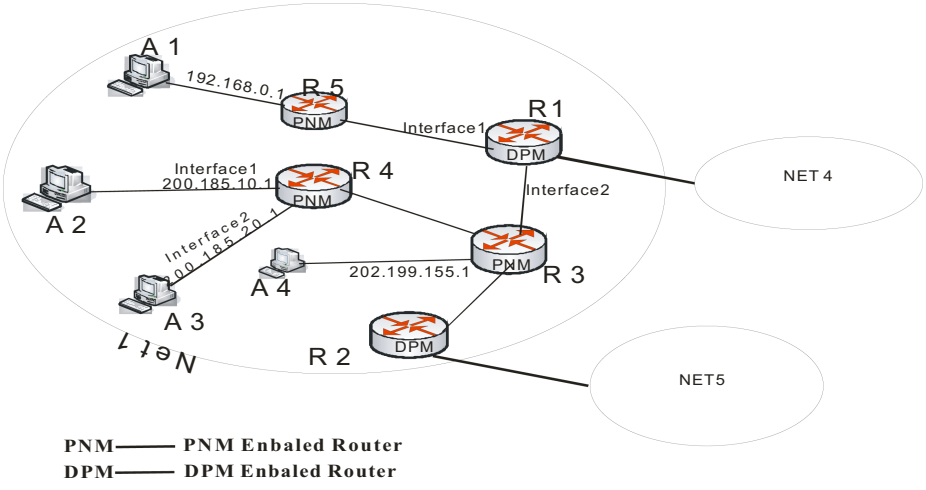


Fig. 3. Deterministic Packet Marking Based Path Numbering (PN-DPM)

incoming interface; PNM enabled routers which are closest to the source of the packet mark each packet traversing them with the path identifiers representing the pathes which link them and the correspongding DPM-enabled routers. The path identifier is obtained by a path numbering algorithm (we discuss more details in section path numbering).

To allocate enough space in IP header for storing the mark, the 16-bit Identification field, the 14-bit Fragmentation related fields, and the reserved 1-bit Flag in the IP header is chosen to be overloaded, when it is certain that a packet is not a fragment or DF is set. Issues related to the overloading of this field have been studied and reported [7]. For fragmented packet, instead, Option field is used [8]. A new sub-section in option field with a new flag for IP traceback was shown in Fig.4.

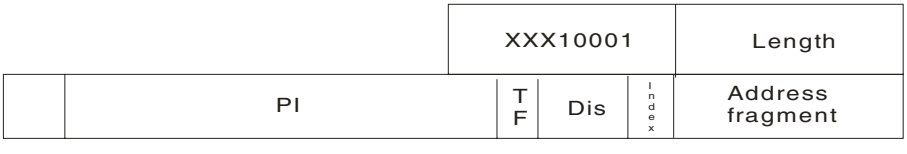


Fig. 4. Fragmented packet mark in Option field

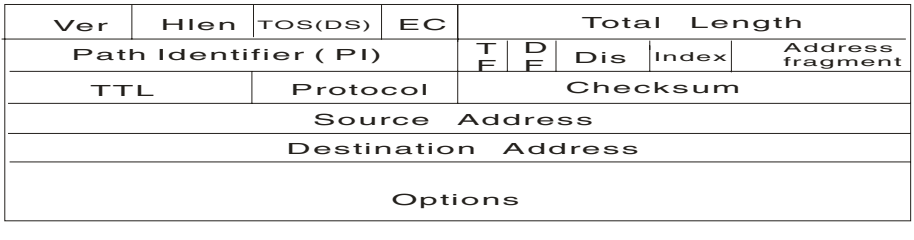


Fig. 5. Un-fragmented packet mark in IP header fixed part

Total overhead for the new sub-section is six bytes, as shown in Fig.5. In our scheme, the 16-bit Identification field is used to store the path identifier. For final receiver knowing if the packet is not fragmented, the DF (do not fragmentation) keep as it old value. The one bit at left side of DF is used for marking flag. The 14-bit Fragmentation related fields, including 1-bit MF (more fragment) and 13-bit offset field, are divided into three sub-fields. The first sub-field is 4-bit long and is used to store the value of distance. It is believed that 4 bits are sufficient since a large domain can be divided into some small domains readily in our scheme. The last two sub-fields are respectively 2-bit index and 8-bit address fragment of the IP address.

3.3 Path Numbering

The work of path numbering is inspired by path fingerprint filtering scheme [5]. In our scheme, path numbering can provide efficient authentication of routers' markings. Moreover, it can be seen as an enhanced path fingerprint, which can be treated not only as a filtering identifier but also as a clue to further traceback.

Step 1: Authentication with CRC-CCITT

CRC is a simple and widely used error-detecting technique in data transmitting. It uses the redundant data of the received data to check if some errors have been introduced during data transmission. CRC-CCITT is one industry standard of CRC. In our scheme, the net-id (NA) of the interface of the PNM enabled router, which is closest to the source of the packets; the incoming interface's IP addresses of the corresponding DPM enabled routers(IA); and the distance (D), from the PNM enabled router ,which is closest to the source of the packets, to the corresponding DPM enabled router are treated as the data being transmitted. The redundant data of the combination of these datas, which is produced by CRC-CCITT, was treated as the path identifier. Each incoming interface of PNM-enabled router is assigned some numbers according to the outgoing net. When traverses the path, the packet will be marked with the number of the PNM-enabled router as the path identifier. Due to the static property of the router numbers and small amount of paths in a marking domin, we can compute the numbers of routers and assign them to PNM-enabled router in advance. When victims receive the marked packets, they can compute to check the integrity of the mark using CRC-CCITT, after reconstructing ingress IP address.

To show the procedure of path numbering, a concrete example will be demonstrated. In Fig.3, we assume that one ingress address (interface 1)of R1 is $IA_1(R1)$, the other (interface 2) is $IA_2(R1)$, the ingress address of R2, IA(R2). According to CRC-CCITT, the divisor's Polynomial is $P(X)=X^{16}+ X^{12}+ X^5+1$, and the function, $HCRC(Y)=Y*2^{16}/P(X)$, is a binary modulo 2 division. In our scheme, the identifier of the path is

$$PI(I_i(Rx1), I_j(Rx2))= HCRC((IA_i(Rx1)*2^n+NET(IA_j(Rx2)))* 2^4+D) \quad (1)$$

Where $I_i(Rx)$ is the incoming interface i of router Rx, $IA_i(Rx)$ is the IP address of $I_i(Rx)$, NET(X) is the net-id of the IP address X, and n is the bit number of NET(I(Rx2)). For example, for NET(I(R5)), n=24.

Then we compute the numbers of the incoming interface (i) of the PNM enabled routers, $N_{i,j}(Rx)$.

$$N_{i,j}(Rx)= PI(I(DPM_j(Rx)), I_i(Rx)) \quad (2)$$

Where $I_i(Rx)$ is the i incoming interface of Rx, $I(DPM_j(Rx))$ is the incoming interface of the DPM enabled router linked to Rx, of which one outgoing interface is linked to net j.

So each incoming interface of PNM enabled routers has some numbers representing the paths. When it is going to mark a packet, it should choose one of them according to the path. The path information can be acquired by the destination address of the packet and the router's routing table. For example, a packet traverses from the source A2 to the destination net4 across R4-R3-R1. According to net4 and the routing table, the incoming interface 1 of R4 will mark the packer with $N_{1,4}(R4)= PI(I_2(R1), I_1(R4))$.

Step 2: Using Time-Released Key Chains

Although step 1 can provide a certain of authentication, it is incomplete for that path identifier and ingress addresses can be forged together in DPM enabled routers and their downstream routers. Then the victim can't distinguish between true and spoofed marks. To solve this problem, we encrypt the path identifier by using the time-released keys authentication scheme. A similar scheme was proposed by Perrig et al. for multicast source authentication [9]. So the numbers of the PNM enabled routers in our step1 are encrypted.

This scheme can be viewed as that each PN-enabled router shares a secret key with each potential victim. The victims can download the keys of the corresponding routers for the latest time interval according the ingress IP addresses from the ISPs' website, and then it is able to compute all the keys for previous time intervals. So path identifiers can be decrypted to authenticate the ingress address.

3.4 Formal PN-DPM Description

In this section, we introduce the formal pseudo code for PN-DPM. As seen from Fig.6 and Fig.7, the path identifier marking procedure is presented as follows. When a IP packet traverse from a host, the incoming interface of PNM-enabled router closest to the source of the packet sets the packet's distance field and the TF-flag to 0. Besides these operations, it set the path-identifier field to its own corresponding encrypted number, according to the packet's outgoing net. Afterwards, if the TF-flag is unset, each PNM-enabled router increases the distance field. Otherwise, nothing does to the packet. Whenever a DPM-enabled router receives an IP packet, it first examines the TF-flag field. If it is 0, the receiving router sets the TF-flag bit to 1, generate a small random number from 0 to 3 as the value of the index field, and insert the corresponding address part mark into the address fragment field.

In the recovery procedure, the PI and distance field can be seen as the digest field in DPM [2].So the similar procedure of ingress address reconstruction is presented in our scheme.

```
1: let p denote an incoming IP packet
2: let p.pi, p.tf, p.df, p.dis, p.index, and p.af denote the PI, Dis, index
   and Addres Fragment field in packet p's IP header, respectively.
3: if the PNM enabled router R is closest to the source of p then
     if p.df = =1 then
        if p's incoming interface is i, outgoing net is net j
        then {p.pi= Ni,j(Rx) ; p.tf=0; p.dis=0;}
     else
       if (option field has enough free space)
           {create a PN - DPM Option structure;
            fill corresponding mark field in option structure;
            fill option field;
            fill length field;
            append the structure to Option field;}
       else
          send a specified ICMP to destination side;
 else
    if p.tf = =0 then p.dis=p.dis+1;
 return;
```

Fig. 6. PNM Enabled Router's Mark Inserting Algorithm

```
1: let p denote an incoming IP packet
2: let p.pi, p.tf, p.df, p.dis, p.index, and p.af denote the PI, Dis, index
   and Address Fragment field in packet p's IP header, respectively.
3: let R and IAj(R) denote a DPM enabled router and i corresponding
    part of its ingress address
4: if p.tf= =0 then generate a small random number j from 0 to3;
     If p.df= =1 then
        {p.tf=1; p.index=j; p.af= IAj(R); }
     Else
       If (option field has a PN - DPM Option structure) then
          fill corresponding mark field in option structure;
       else
          send a specified ICMP to destination side;
return;
```

Fig. 7. DPM Enabled Router's Mark Inserting Algorithm

4 Discussions and Analysis

As well-know, A CRC is always used in the same way as a checksum to detect accidental alteration of data during transmission or storage. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. However, CRCs cannot be safely relied upon to fully verify data integrity in the face of intelligent (rather than random) changes for the linear relation between the CRC redundancy data and the data transmitted. An effective way to protect messages against intentional tampering is by the use of a message authentication code such as HMAC.

In our scheme, CRC-CCITT can be merely seen as a hash function H that the input is IP addresses IA , net-id NA ,and the distance D, and the output is PI and D.

$$(PI,D)=H(IA,NA,D)$$

$$PI=CRC\text{-}CCITT(IA,NA,D) \quad (3)$$

Because the false positive of CRC-CCITT is near the expected value $1/2^{16}$, and the total expected value of false positive of H in our scheme is $1/2^4*2^{16}$, for the effect of the distance field. So, it is effective. The practical work of authentication is provided by using the time-released keys authentication scheme which can be seen as a type of HMAC. The linear relation between the (IA,NA,D) and PI introduced by the CRC-CCITT, can be eliminated largely by a elaborately chosen HMAC function.

The number of attackers, which PN-DPM can traceback, with the false positive rare limited to 0, is evaluated. Let us examine the origin of false positive. If there is only one ingress address with a given digest, there will be no false positives because of the properties of our proposed hash function. Therefore, the rate of false positives is nearly 0 when the number of attackers, A, is less or equal to the number of possible digests, 2^{20}. Another important consideration is the expected number of packets required for the reconstruction. This number will be related to n, the number of segments that the ingress address was split. The expected number of packets, E[N],required to be marked by a single DPM-enabled interface in order for the victim to be able to reconstruct its ingress address is given by a Coupon Collector problem discussed in [10],

$$E[N]=n(1/n + 1/n\text{-}1 + 1/n\text{-}2 + \ldots + 1) \quad (4)$$

In our scheme, the required number is 8, which is very small.

5 Conclusion

In this paper, we have presented a new deterministic packet marking scheme for IP traceback. Through a unique technique: path numbering, our scheme effectively addresses shortcoming of existing DPM in some degree. And our scheme has significantly good performance in dealing with large-scale DDOS attacks as a IP traceback technique by the above analysis. In addition, the path identifier can be used directly as a path fingerprint to detect and filter various types of attacks by the victims, not only for DOS or DDOS attacks since the number of packets required is small. Furthermore, the more accurate information of the path identifier, PI, can be provided as a value-added service by ISPs for post-mortem traceback, that may strongly motivate ISPs to deploy the scheme owing to the direct benefit.

References

[1] Belenky, A., Ansari, N.: IP Traceback with Deterministic Packet Marking. In: IEEE Communications Letters, pp. 162–164 (2003)

[2] Belenky, A., Ansari, N.: Tracing multiple attackers with deterministic packet marking (DPM). In: Proc. IEEE Pacific Rim Conf. Commun.,Comp. and Sig., pp. 49–52. IEEE Computer Society Press, Los Alamitos (2003)

[3] Belenky, A., Ansari, N.: Accommodating fragmentation with deterministic packet marking for IP traceback. In: Proc. IEEE GLOBECOM, pp. 374–1378. IEEE Computer Society Press, Los Alamitos (2003)

[4] Gao, Z., Ansari, N.: Tracing cyber attacks from the practical perspective. IEEE Commun. Mag., 123–131 (2005)
[5] Fu-Yuan, L., Shiuhpyng, S.: Denfending against spoofed DDoS attacks with path fingerprint. Computers & Security 24, 571–586 (2005)
[6] Sung, M., Xu, J.: IP traceback-based intelligent packet filtering: a novel technique for defending against internet DDoS attacks. IEEE Transactions on Parallel and Distributed Systems 14(9), 861–872 (2003)
[7] Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Network support for IP traceback. IEEE/ACM Trans. Networking, 226-237 (2001)
[8] Dae-Sun, K., Choong-eon, H., Yu, X.: An Intelligent Approach of Packet Marking at Edge Router for IP Traceback. Lecture Notes in Artificial Intelligence, 303–309 (2005)
[9] Perrig, A., Canetti, R., Song, D., Tygar, D.: Efficient and secure source authentication for multicast. In: Network and Distributed System Security Symposium (2001)
[10] Feller, W.: An introduction to Probability Theory and its Applications, 3rd edn. Addison-Wesley, Reading (1968)

A Designing Method for High-Rate Serial Communication*

Gongxuan Zhang, Ling Wang, and Bin Song

School of Computer Science & Technology, Nanjing University of Science & Technology
210094 Nanjing, China
{gongxuan,arch601,face601}@mail.njust.edu.cn

Abstract. In this paper, after briefly introducing the characteristics of MSP430F149 MCU, we present the design of a new high speed serial communication method based on MSP430F149. The new method can correctly identify the parity bit without the knowledge of the parity mode (odd/even/address bit). We provide the relevant design of both the hardware and the software. This serial communication scheme can support speeds up to 1.5Mbps.

1 Introduction

Serial communication is widely used in MCU-based industrial control systems. The serial communication rate of most MCU is usually below 1Mbps. However, some modern systems often demand serial communication with higher rates. For instance, one elevator data collection system designed by the authors requires a rate of 1.5Mbps. In addition, the parity mode of the elevator is unknown. To collect the elevator's data, the TSHBIA CV150 elevator is connected to the data collection system through a RS480 interface. The collected data is processed and send to other following systems through a RS232 interface. Therefore the MCU is required to support a serial communication rate of 1.5Mbps. MSP430F149 MCU provides two USARTs. In this paper, we shall present the software and the hardware design of the system to meet the 1.5Mbps serial rate requirement.

The rest of the paper is organized as follows. In section 2 MSP430F149 Architecture is presented. In section 3, the Hardware Design for High-Rate Serial Communication is presented. Section 4 presents Software Design for High-Rate Serial Communication. Conclusion is given in section 5.

2 MSP430F149 Architecture

MSP403F149 MCU is a member of the latest family of 16-bit ultra-low power consumption MCUs provided by Texas Instruments Inc. It can be supplied by a low and wide power voltage range from 1.8V to 3.6V. It provides 60KB FlashROM, 2KB RAM, and supports serial on-chip programming. This provides the flexibility for

* This is supported by 'National 863 Plan', No. 2006AA01Z447.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 518–524, 2007.

users to compile programs and to control design parameters. MSP403F149 can be erased up to 100,000 times and has a strong protection against interference.

MSP430F149 comes with powerful interrupt functionality. It provides multiple registers for output, function selection, and interrupts, which enables the reuse of functional ports and universal I/O ports – The function of an I/O port needs to be specified before it can be accessed. This mechanism enhances the ports' functionality and flexibility. MSP430F149 has two universal synchronous/asynchronous receive/transmit communication interfaces (USART). It also has a convenient development and debugging environment with embedded JTAG debugging interface, which only requires a PC and a JTAG debugger to develop. The developing languages include C language and Assembly language.

3 Hardware Design for High-Rate Serial Communication

Figure 1 illustrates the serial communication scheme for the elevator data collection system designed by the authors. The system's serial rate is 1.5Mbps. The hardware design shares some common features of a typical serial communication system. Port P1.0 is used to control the transmission and reception of MAX485. In the following sections, we shall focus our discussion on those features in Figure 1 that are particularly designed to support a 1.5Mbps serial communication rate.

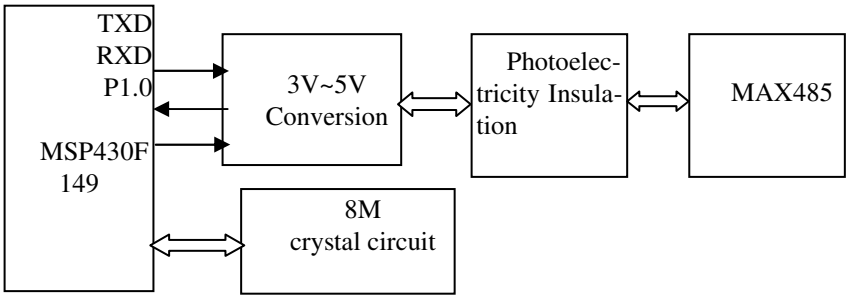


Fig. 1. The high-rate serial communication scheme

3.1 3V – 5V Voltage Level Conversion

There exist several methods to implement the 3V-5V voltage level conversion. For instance, we could use PHILIPS 74LVC4245 octal dual supply translating transceiver chip and some simple voltage transition chip such as 74LVC07. We use a different voltage conversion circuit as shown in Figure 2. The key of the design is to insure that all the circuits have fast transition speeds. As shown in Figure 2, Port TXD0 and P1.0 of MSP430F149 are output ports which can support the conversion from 3V to 5V when connected to certain pull-up resistance. For the input port RXD0, a proper choice to support the conversion from 5V to 3V is to use 74HC05 OC gates. The pull-up resistance R3 and R4 are connected to 3.3V and 5V, respectively with values

around 1K ohms. If their values are too high, it is hard for 74HC05 to pull up to high output voltage, especially under high-speed voltage transition.

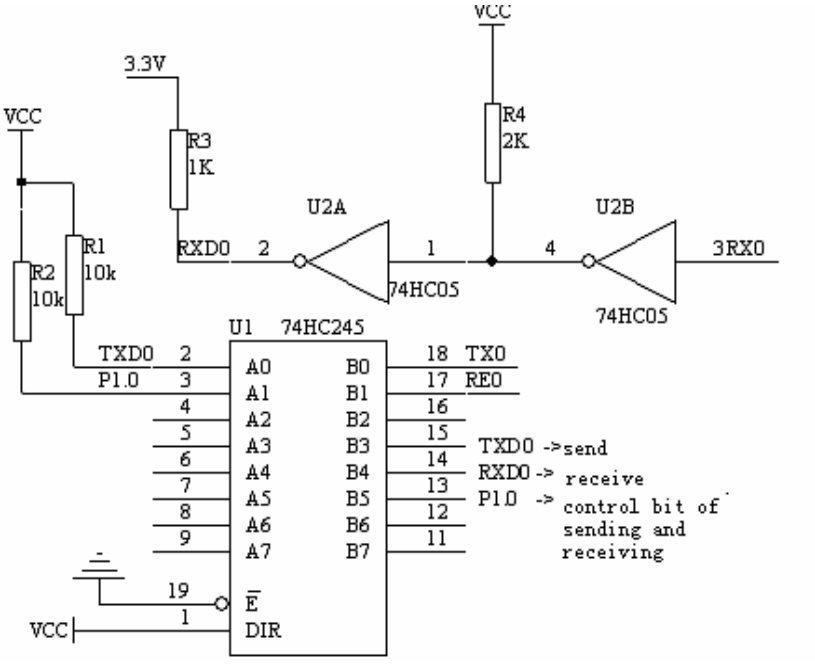


Fig. 2. The transition voltage circuit of 3V~5V

3.2 Crystal Frequency Selection

In the ideal case, the crystal frequency should be an integer times of the serial communication rate. However, this is not an absolute rule. The crystal frequency ought to be selected with respect to the specific design scenario. Some articles show that the crystal of MSP430 can reach 8MHz up to 10MHz with a smallest frequency division factor of 3. Given that the serial rate in our system is 1.5Mbps, we used 6MHz crystal frequency at the beginning. The source clock of the baud rate generator is SMCLK whose frequency is also set to 6MHz with a frequency division factor of 4. Thus, with a character length of 8 bits and with parity check, the serial port receiving program has a time window of 44 MCLK to process each received character.

As we know, MSP430 MCU is based-on RISC, so some instructions are simulated. The average period of the simulated instructions is 4 MCLK. That means the serial receiving program can only use about 10 instructions to process a single character. This imposes a fairly tight time constraint. Through repeated experiments with crystal frequencies from 6MHz to 12MHz (the frequencies might exceed the normal level, but can still run in this system), we eventually selected 8MHz as the final choice. With the 8MHz clock, the frequency division factor is not longer an integer. So we

need to set the baud rate selector properly. See Section 4 for the detailed discussion on setting the baud rate selector.

4 Software Design for High-Rate Serial Communication

The tasks of software design for this high-rate serial communication system are: 1) Initializing the corresponding registers, 2) Receiving the data coming from the serial port in a timely manner, 3) Decoding the parity bits and saving the data. The emphasis of our discussion on software design is on completing the above tasks within the time of 44 MCLK (8MHz).

4.1 Initialization of Serial Port Registers

The key of serial communication is to set the baud rate selector0、baud rate selector1 and baud rate modulate controller UMCTL. With an 8MHz crystal and a 1.5Mbps serial rate, the frequency division factor is about 5.333333. So the value of baud rate selector0 is 5, and the value of baud rate selector1 is 0. The method to set the UMCTL is shown in Table 1.

Table 1. UMCTL Set

Fraction addition	Carry to next integer	UMCTL	Bits
0.333333+0.333333=0.666666	no	m0	0
0.666666+0.333333=0.999999	no	m1	0
0.999999+0.333333=1.333332	yes	m2	1
1.333332+0.333333 =1.666665	no	m3	0

In the same way, we can conclude that the value of UMCTL is 24H. The initialization of serial port program is provided below.

```
/*********************
        serial port initialization function
             ***********************/
 void InitUsart1(void)
 {
  unsigned char i;
  WDTCTL = WDTPW + WDTHOLD;// stop watch dog
  BCSCTL1&=~XTOFF;          // use high crystal XT2
  do{
   IFG1&= ~OFIFG;         // clear OSCFault flag
   for (i = 0xff; i > 0; i--);
    }while ((IFG1 & OFIFG) == OFIFG);  //waiting for OS
                                       Fault flag reset
   BCSCTL2 |= SELM1+SELS;   // the clock source of MCLK
                               and SMCLK is XT2
   UCTL1&=~SWRST;
```

```
    UCTL1 = PENA+CHAR;    // define character length and
                                  parity mode
    UBR01 = 0x05          // define baud rate selector0
    UBR11 = 0x00;         // define baud rate selector1
    UMCTL1 = 0x24;        // define baud rate adjust con-
                             trol register
    UTCTL1 = SSEL1;      // the clock source of baud rate
                            generator is SMCLK
    URCTL1=URXEIE;        // allow receive erro
    ME2|=UTXE1+URXE1;     // allow receive and send
    P3SEL |= 0xC0;      // P3.6 P3.7 is TXD/RXD of USART1
     P3DIR |= 0x40;
   }
```

4.2 Serial-Port Data and Decoding of the Parity Bit

To identify parity bit, we can use any parity mode to receive data, for example the odd mode, and save the parity error state (PE) when receiving the character. This way the following processing program can ascertain the parity bit value in terms of the number of 1's in each character and the PE state. To improve the realtimeness of the data processing and the flexibility of control, we combine C language and Assembly language in our programming.

```
  /*****************************************/
  /*main function*/
  /*****************************************/
 #define ReceiveCount   100
 #include <msp430x14x.h>
  extern  void  re_tx(unsigned  char*pbuff,unsigned  int
re_count);
   /*serial  receiving  function  (assemble  function)—
prototype announce*/
   extern unsigned char judge_odd_even(unsigned char test-
char);
  /*The  function  of  counting  number  of  1  of  character
(assemble function)*/
   void  trans_re(unsigned  char  *pbuff,unsigned  int
trans_count);
 unsigned int re_count=ReceiveCount;
 unsigned char buffer[2* ReceiveCount];
 void main( void)
   {   InitUsart1()•
     re_tx(buffer,re_count);   /*call serial receive func-
tion\*/
     trans_re(buffer,re_count);/*call parity identify fun-
tion*/
     }
     ;****************************************
     ; filename: usart1.s43  serial receive function
     ; receive specified character by querying mode
         NAME usart
```

```
        #include <msp430x14x.h> ;head file
        PUBLIC re_tx1
        RSEG CODE  ; code is relocated
    re_tx1
        bit.b #URXIFG1,&IFG2                ;5
        jz re_tx1       ;not receive, loop  ;2
        mov.b &URCTL1,1(R12)

        bic.b #PE+FE+OE+BRK+RXERR,&URCTL1
        bit.b #PE,&URCTL1                   ;5
       jz jnoerror  ; no error             ;2
        bic.b #PE+FE+OE+BRK+RXERR,&URCTL1   ;5
        mov.b #00h,1(R12) ; error #00h      ;5
        jmp jnext                           ;2
    jnoerror
       mov.b #0FFh,1(r12)    ; right #FFH  -->;5
    jnext
        mov.b &RXBUF1,0(R12)                ;6
        incd.w r12                          ;1
        dec.w  r14                           ;1
        jnz re_tx1                          ;2
        ret
        end
    ;*****************************************
    ;filename: odd_even.s43  the function of counting the
number of 1 of character
    ;
    NAME judge_odd_even
                #include <msp430x14x.h>
                PUBLIC judge_odd_even
                RSEG CODE
    judge_odd_even
                mov.b r12,r13
                clr.b r12
                mov.b #08h,r14
    loop_rra    rra.b r13
                jc xorr12
                jmp jnext
    xorr12      xor.b #0ffh,r12
    jnext       dec.b r14
                jnz loop_rra
                ret
                end
    void  trans_re(unsigned  char  *pbuff,unsigned  int
trans_count)
    {
      unsigned int test_count;
      unsigned int odd_even;
      unsigned int twice_count;

for(test_count=0;test_count<trans_count;test_count++)
```

```
       { twice_count=test_count+test_count;
         odd_even=judge_odd_even(pbuff[twice_count++]);
         if(odd_even==odd)// the number of 1 is odd
         {

if(pbuff[twice_count]==0xff)pbuff[twice_count]=0x00;
           else
if(pbuff[twice_count]==0x00)pbuff[twice_count]=0xff;
           else{}
         }//if
         else
         {
         }//else
      }//for
  }
```

5 Conclusion

Without the knowledge of the parity mode, the proposed high-rate serial interface can successfully receive data at 1.5Mbps when connected to TOSHIBA CV150 Elevator system. It meets the system design requirement and provides good performance.

References

1. Hu, D.: The Super-low Energy Resume MCU of MSP430 Series, Beijing University of Aeronautics and Astronauatics Press, Beijing (2001)
2. Wei, X.: The Interface Technology and System Design Instance of MSP430 Series, Beijing University of Aeronautics and Astronauatics Press, Beijing (2002)
3. TI Corporation: Mixing C and Assembler With the MSP430 Application, Beijing (2002)

A Comprehensive Efficient Flooding Algorithm Using Directional Antennas for Mobile Ad Hoc Networks

Xianlong Jiao, Xiaodong Wang, and Xingming Zhou

PDL, National University of Defense Technology, Changsha, China, 410073
jiaoxianlong1982@gmail.com, {xdwang,xmzhou}@nudt.edu.cn

Abstract. Flooding is often used as a building block for route discovery in routing protocols for mobile ad hoc networks. The traditional implementations of flooding suffer from the broadcast storm problem. To solve the broadcast storm problem, some efficient flooding schemes have been proposed. However, there also exist many problems in these schemes, such as signal collision. In this paper, we propose a comprehensive efficient flooding algorithm called CEF to extend our previous work in [1]. We attempt to solve the problems which are not solved by our previous proposed algorithm. We have implemented our algorithm in ns-2 simulator, and the results show that our algorithm achieves better performance than Pure Flooding, EF1 (the algorithm proposed in [2]) and our previous proposed algorithm due to smaller number of signal collisions.

Keywords: Flooding, Directional Antenna, Ad Hoc.

1 Introduction

Mobile ad hoc networks are consisted of mobile nodes with limited bandwidth, computing ability and energy, which is different from the traditional wired networks. Many routing protocols such as TORA [3], AODV [4] and DSR [5] are proposed for this kind of networks. Network wide flooding is an important utility function in mobile ad hoc networks, which tries to send packets from a source node to all other nodes in the ad hoc networks, so it's often used as a building block for route discovery in on-demand ad hoc routing protocols.

In [6], the author introduces the broadcast storm problem which is raised when all nodes relay the received packet after they receive it. This will cause contention and collision in the shared wireless channel. To solve the broadcast storm problem, some efficient flooding schemes have been proposed such as [7], [8] and [9]. The author of [10] compares various flooding techniques in mobile ad hoc networks, and classifies these techniques into four categories: Simple flooding, Probability based methods, Area based methods, and Neighbor knowledge based methods.

In this paper, we propose a comprehensive efficient flooding algorithm called CEF to extend our previous work EFDA in [1]. EFDA is based on EF1 algorithm (the algorithm proposed in [2]). We assume that every node is provided with a single-beam directional antenna in EFDA. In CEF, we attempt to solve the problems existing in EFDA and aim to achieve better performance such as the number of collisions, delivery rate and so on.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 525–534, 2007.

The rest of the paper is organized as follows. In Section 2, we give an overview of the related works concerning flooding techniques and directional antennas. Our algorithm is described in Section 3. In Section 4, we evaluate the algorithm and give the detailed results. Finally, we present the conclusion and future work in Section 5.

2 Related Works

Many works have been researched on flooding protocols. One of the notable works is [2], in which Hai Liu et al. proposed an efficient flooding scheme based on 1-hop information in mobile ad hoc networks. Their algorithm achieved the local optimality in two senses: 1) the number of forwarding nodes in each step is the minimal; 2) the time complexity for computing forwarding nodes is the lowest, which is O(nlogn), where n is the number of neighbors of a node. They compared their algorithm with three deliverability-guaranteed schemes through simulation: Pure flooding, Edge Forwarding (it requires 1-hop information) [7], and CDS-based flooding [9] (it requires 2-hop information). Comparison of one of the performance (that is the number of collision) encourages us to consider utilizing the directional antenna to further optimize the flooding algorithm.

Recently directional antenna has been exploited in many papers for optimizing the performance. In [11], Akis Spyropoulos et al. demonstrated the benefits of using directional antennas in ad hoc networks. They presented an energy-efficient algorithm for routing and scheduling in ad hoc network with nodes using directional antennas. Their algorithm can decrease the total energy consumption and thus increase network lifetime by a factor, which is proportional to the antenna gain. The authors of [12] evaluated the tradeoffs involved in using directional antennas in ad hoc routing. They evaluated the performance of DSR using directional antennas. They identified several issues that emerged from executing DSR over directional antennas. Their analysis showed that by using directional antennas, ad hoc networks might achieve better performance. Jian Tang et al. studied several interference-aware routing problems in multihop wireless networks using directional antennas in [13]. They have presented new definitions for the link and path interference that are suitable for designing better routing algorithms.

Other work concerning directional antenna includes [14], [15], and [16]. As what is discussed in [14], use of directional antennas allows concentration of the beam toward the intended destination without wasting energy in unwanted directions. Further, because the beam is generated only toward a certain direction, it creates less interference to other nodes that are outside the beam, which enables greater information capacity in the network. Much work is exploiting directional antennas for optimizing energy consumption, and we consider using directional antennas for optimizing the signal collisions in this paper. There are three components for a directional antenna: beam-radius, beam-width and beam orientation. For simplicity, we fix the beam-radius as the transmission range and only adjust the beam-width and beam orientation to achieve good performance.

3 Comprehensive Efficient Flooding Using Directional Antennas

3.1 System Model

In our system, a mobile ad hoc network with N nodes is considered. These nodes are distributed over a specific area, and are equipped with a directional antenna for transmission. There are three parameters which are related to the directional antenna: the beam radius, the beam-width, and the beam orientation. Each node can adjust the beam-width and beam orientation of the directional antenna. For simplicity, we assume that the beam radius is identical for each node, and is fixed as the transmission range of the wireless node. Furthermore, F(S) represents the forwarding nodes set of S, and N(S) describes the neighbor nodes set of S.

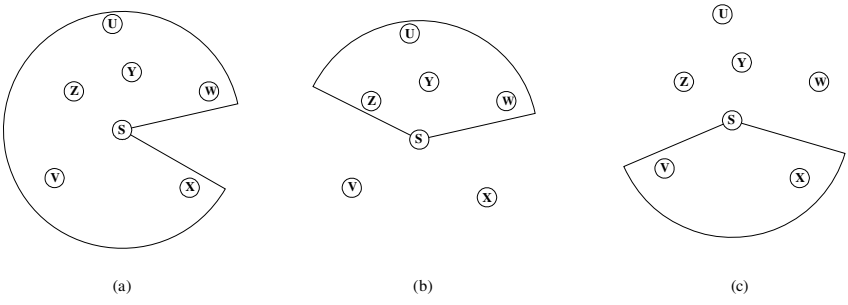


Fig. 1. Examples of different beam-widths and beam orientations

When a node adjusts beam-width or beam orientation, the coverage area of the transmission signal can be different. As can be seen in Fig. 1(a), node S can cover all the nodes when the beam-width is large enough. In Fig. 1(b), S decreases the beam-width, and only covers Z, U, Y, W. If S changes the beam orientation, the coverage area will be similar to the one shown in Fig. 1(c).

3.2 Further Modified Optimized Forwarding Node Selection Algorithm

In this section, we detail the further modified optimized forwarding node selection algorithm. The boundary merge algorithm and the forwarding node selection algorithm which is used in EF1 is directly exploited in our algorithm without any change.

EF1 optimizes the forwarding nodes set through removing from F(U) the nodes covered by $\{S\} \cup \{V | V \in (F(S) \cap N(U))$ and $id(V) \leq id(U)\}$ (defined as node-set (U)). In our algorithm, when we use this method to optimize the forwarding node set, we also determine the beam-width and beam orientation of the directional antenna. Consider the example given in Fig. 2(a), and assume that F(S) = {U, V, W}. In Fig. 2(a), we assume that id(U)<id(V) and id(U)<id(W). We first define a subset of N(U) as NNC(U), which contains the neighbors of U which are not covered by node-set (U). J and D are in the overlapping area, and the id of U is the smallest. Therefore, NNC(U)={J, A, C, D}, while NNC(V)= {B, I} and NNC(W) ={F, G, H}. Through comparing the angles formed by S, U and the nodes in NNC(U), we can determine the

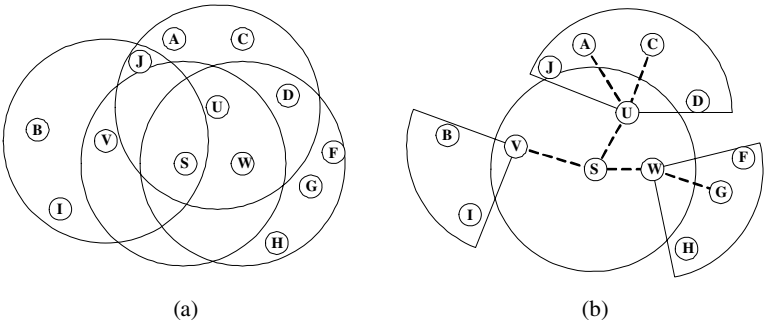


Fig. 2. Examples for determining the beam-width and beam orientation of directional antenna

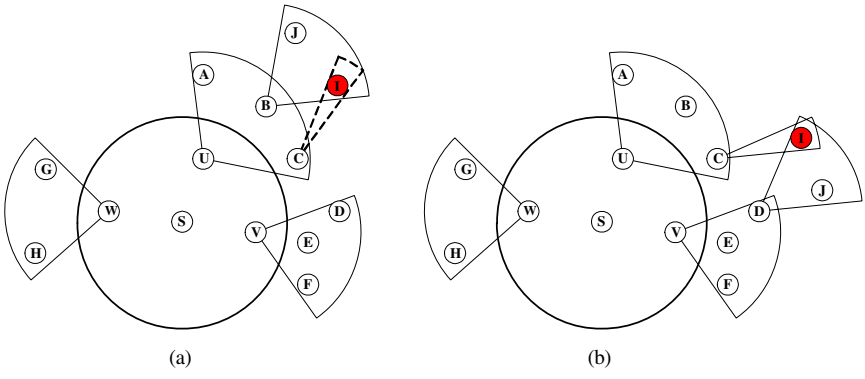


Fig. 3. Problems in EFDA

two sides of the sector. For example, the angles formed by S, U, and NNC(U) are $\{\angle SUD, \angle SUC, \angle SUA, \angle SUJ\}$. We order these angles in non-decreasing order according to their sizes. If there are two angles which are equal in size, then we put forward the angle whose id is smaller. After ordering these angles, we can find the smallest angle $\angle SUD$ and the biggest angle $\angle SUJ$. Therefore two sides of the sector can be determined through this method. As there are two sectors which match the condition, we need to further determine the right sector. Since the forwarding node should forward the message to the uncovered area, the sector should not include node S. So we can further determine the beam-orientation and the beam-width of the directional antenna as shown in Fig. 2(b).

The optimization discussed above is adopted in our previous work in [1]. However, there are also some problems in the optimization, as we only consider the signal collision existing in the 2-hop area. Take Fig. 3 as an example to explain the problem. In Fig. 3(a), we assume that F(S)={U, V, W}, F(U)={A, B, C}, F(V)={D, F}, and F(W)={G, H}. The signal collision at node I can be avoided, as B and C are both the neighbors of node U, and they will receive the message attached with F(U)={A, B, C}. Since $B \in (F(U) \cap N(C))$ and $id(B) \leq id(C)$, C will not transmit the message to node I. Nevertheless signal collision will probably occur at node I in Fig. 3(b).

FModOptFwdNodes Algorithm

Input: message m from S.
Begin
if m was received before, **then** discard m;
else
Deliver m to upper layer;
if this node, say U, is in forward-list in m
Compute F(U);
Remove from F(U) the nodes that are covered by node-set(U);
Order the angles formed by S, U and U's neighbors in NCS(U)(these neighbors are in NNC(U) and choose U as the source node through the HELLO message) in non-decreasing order, and determine the two sides of the sector according to the smallest angle and the biggest angle;
Choose the sector which doesn't include S, and determine the beam-width and the beam-orientation of its directional antenna;
Attach F(U) to m and transmit m out with the directional antenna.
End

Fig. 4. FModOptFwdNodes Algorithm

Consider node C and node D in Fig. 3(b). For node C, node I is not covered by node-set(C), so node C will further forward the message to node I. For node D, node I is also not covered by node-set(D), so node D also will consider node I for further transmitting messages. Therefore signal collision will probably occur at node I.

Here we present a solution to the problem. As each node periodically broadcast the HELLO message to exchange the information of its ID and its geographic location, we add a tag into the HELLO message to denote that whether a node is a forwarding node. Initially the tag is set as 0. When a node receive the flooding message and find that it's a forwarding node, it change the tag as 1 to denote that it's a forwarding node, add the flooding message ID into the HELLO message, and wait some time to further forward the flooding message. When a node receives a HELLO message from its neighbor, it checks the value of the tag in the HELLO message. If the value is 1, then it examines whether it has received the flooding message before according to the flooding message ID attached in the HELLO message. If it has received the flooding message before, it doesn't change the tag value of its HELLO message. If it hasn't received the flooding message before, since it may receive multiple HELLO messages with the value of tag as 1, it chooses the node whose ID is the smallest, adds the node ID into the HELLO message, and changes the tag as 2 to denote that it has chosen a source node to transmit the flooding message to it. Then it changes the value of the tag back to 0 when it's ready to send the next HELLO message. So if a node finds that the value of tag in the received HELLO message is 2, it checks whether it is the chosen source node. It can further determine the nodes it should transmit the flooding message to. As shown in Fig. 3(b), node I will receive the HELLO messages coming from node C and node D with the value of tag as 1(since node C and node D are both forwarding nodes). Node I will choose node C as the source node, so D is only in charge of sending the flooding message to node J. Therefore signal collision at node I can be avoided. Since the information about the beam-radius and the beam orientation

can be maintained for the next flooding, the operations discussed above are only needed at the beginning for initialization. We will discuss the situation when nodes move in next section.

We further modify the ModOptFwdNodes algorithm proposed in EFDA which is based on the optimized forwarding node selection algorithm proposed in EF1. It is incorporated with the directional antenna characteristic and the HELLO message. Fig. 4 shows the further modified optimized forwarding node selection algorithm FModOptFwdNodes. NCS(U) is a subset of NNC(U), which removes the nodes from NNC(U) which don't choose U as the source node. The added operations are written in italic type. When a node receives a flooding message, this algorithm will be invoked.

3.3 Further Modified Topology Update Algorithm

When nodes move in mobile ad hoc networks, the neighbor relation may change, and accordingly the forwarding nodes set will be different. In [2], the author proposes an efficient algorithm that can incrementally update the forwarding node set as the topology changes. The forwarding node set is maintained at each node and is always ready for use. Here, we must update the beam-width and the beam orientation of the directional antenna when nodes move.

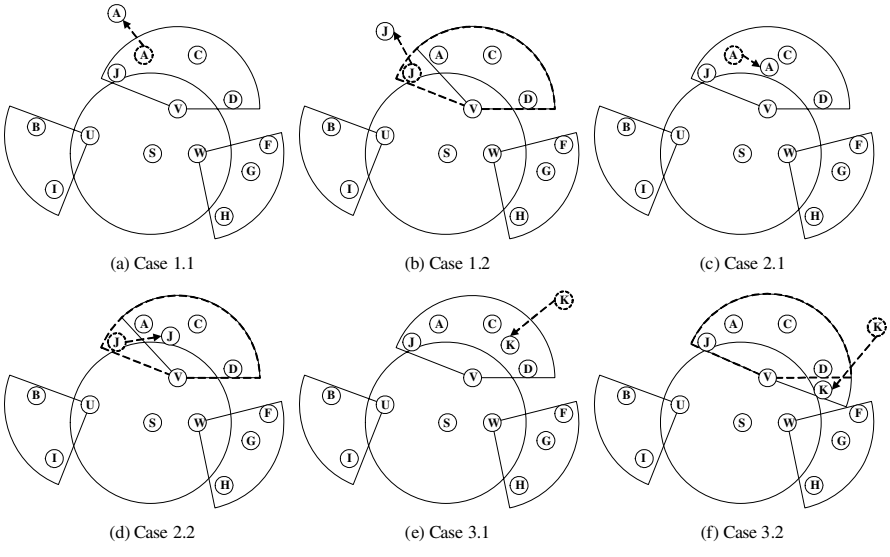


Fig. 5. Cases for determining the beam-width and the beam orientation

There are three cases to consider: 1) a neighbor in NCS(U) moves out of NCS(U); 2) a neighbor in NCS(U) moves but still in NCS(U); 3) a node outside NCS(U) moves in NCS(U). Every case has two sub-cases as shown in Fig. 5

For case 1, if the angle formed by S, U and the neighbor is neither the biggest nor the smallest, then there is no need to update the beam-width and the beam orientation (case 1.1). If the angle is the biggest or the smallest, then find the biggest angle or the

smallest angle among the other nodes in NCS(U) (case 1.2). As we have already ordered these angles, we can acquire these two angles easily. According to the biggest angle and the smallest angle newly computed, we can update the beam-width and the beam orientation of the directional antenna. For case 2, there are also two sub-cases. If the angle formed by S, U and the neighbor is neither the biggest nor the smallest, then we don't update the beam-width and the beam orientation (case 2.1). Handling of case 2.2 that the angle is the biggest or the smallest is similar to handling of case 1.2. There is some difference that we still need to compare the angle formed by S, U and the neighbor which has moved with other angles, since the neighbor is still in NCS(U). For case 3, if the angle formed by S, U and the node is smaller than the biggest angle and bigger than the smallest angle, then no update is needed (case 3.1). If the angle is bigger than the biggest angle or smaller than the smallest, we can update the beam-width and the beam orientation with this angle and the smallest angle (or the biggest angle) (case 3.2). The further modified topology algorithm is shown in Fig. 6.

Further Modified Topology Update Algorithm

Input: V that changes its location to U.
Output: updated F(U), beam-width and beam orientation
Begin
  if V$\notin$ F(U) and V is now in N(U)
    Find arcs in B that are affected by disk V;
    //Suppose k arcs B[i], B[i+1], … , B[i+k-1] are affected.
    BoundaryMerge({B[i], B[i+1], … , B[i+k-1]}, d(v));
  if V$\in$ F(U)
    Find arcs in N(U) that are affected by V's leaving;
    //suppose k arcs are affected.
    Compute the boundary of the affected k arcs;
    Find arcs in B that are affected by V's current place;
    //suppose l arcs B[i], B[i+1], … , B[i+l-1] are affected.
    BoundaryMerge({B[i], B[i+1], … , B[i+l-1]}) are affected.
  Update F(U) based on the new boundary B.
  For cases 1.1, 2.1, and 3.1, there is no need to update the beam-width and the beam orientation.
  Adjust the beam-width and the beam orientation of the directional antenna according to cases 1.2, 2.2, and 3.2.
End

Fig. 6. Further Modified Topology Update Algorithm

4 Performance Evaluation

We use ns-2 simulator [17] which is downloaded from the web site to evaluate our algorithm. VINT project develops ns-2 simulator, and then Monarch extends the simulator for wireless network. We exploit Lucent's WaveLAN with a nominal bit rate of 2Mb/sec. Two-ray ground model is used as the radio propagation model.

Motion follows the random way point model. Directional antennas have been incorporated into this simulator.

Our algorithm CEF is compared with Pure Flooding, EF1, and EFDA. As the forwarding nodes are the same as EF1 and EFDA, we only test two metrics: the number of collisions and the deliverability ratio. To analyze the performance in static and mobile scenarios, we study these two metrics against three parameters: the number of nodes, transmission range, and pause time. These two metrics are defined as follows.

1) the number of collisions: the sum of collisions that each node experience before it receives the flooding message correctly.
2) deliverability ratio: the number of nodes that successfully receive the flooding messages over the total number of nodes in the network.

We consider networks with various sizes (200, 400, 600, 800, 1000) in static and mobile scenarios. These nodes are randomly placed on a rectangular field. The source node is randomly chosen for sending the flooding message. For every size of networks, 100 separate runs are executed and we use the means. Other parameters are shown in Table 1 which is similar to that of EF1.

Fig. 7 shows the number of collisions and deliverability ratio against the number of nodes, transmission range and pause time. From Fig. 7, we can see that the performance of Pure Flooding is significantly decreased with the increase of the number of nodes. The density of the network will increase with the number of nodes. As every node forwards the flooding message in Pure Flooding, a larger number of signal collisions will occur and more re-transmissions will be needed with the increase of the density of the network. It is so called the broadcast storm problem. As a result, some nodes will not receive the message. EF1, EFDA, and CEF all maintain good performance, as the number of forwarding nodes of these three algorithms is the minimal, and the number of signal collisions is smaller than Pure Flooding especially after the number of nodes reaches 600. Furthermore, CEF further consider the problem of signal collision which is not solved in EFDA, so CEF achieves the best performance. When transmission range increases, every node will have more neighbors, and it will increase the probability of signal collision when these neighbors are forwarding the flooding message. So there will be more signal collision with the increase of the transmission range in Pure Flooding. For EF1, EFDA, and CEF, the change is not significant in the curves of the number of collisions, and CEF also

Table 1. Simulation Parameters

Parameter	Value
Mac Layer	IEEE 802.11
Data Packet Size	256 bytes
Bandwidth	2 Mb/s
Transmission Range	100~300 meter
Number of Node	200~1000
Pause Time	10~ ∞ s
Size of Square Area	1,000,000 meter2
Number of Trails	100

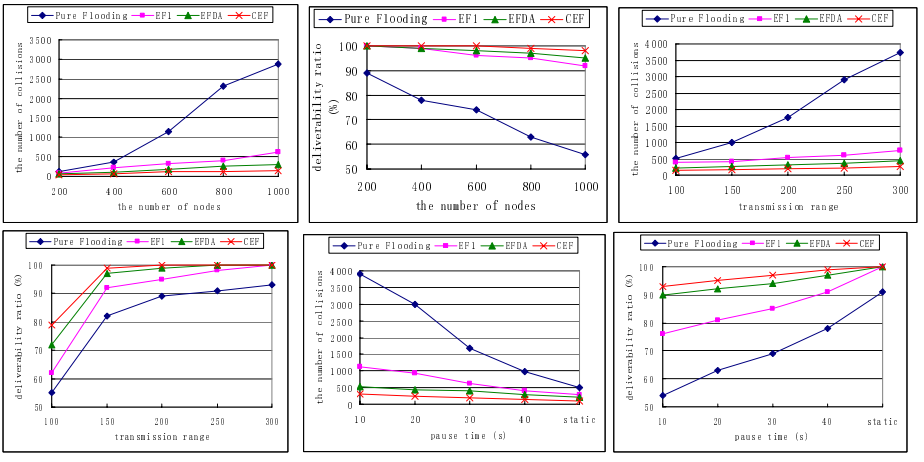


Fig. 7. The number of collisions and deliverability ratio against the number of nodes, transmission range, and pause time

performs the best as a result of the reason discussed before. Notice that deliverability ratio increases with increase of the transmission range in all three curves, as nodes have more chances to receive the flooding message when the transmission range increases. In Fig. 7, we can observe that the number of collisions of Pure Flooding increases significantly with the decrease of pause time. As a node will move more frequently with the decrease of pause time, the neighbors of the node will also change frequently, and the probability of signal collisions will increase. As discussed before, every node will forward the flooding message in Pure Flooding, more signal collisions will occur and more nodes will not receive the message as a result of signal collision and mobility. The number of signal collisions of EF1, EFDA, and CEF is much smaller than Pure Flooding. When nodes move more frequently, the forwarding nodes will change more quickly, and therefore more retransmission are needed due to increase of signal collision. So the deliverability of EF1 drops quickly after pause time reaches 30s, and CEF maintains the best performance due to the smallest number of collisions.

5 Conclusion and Future Work

In this paper, we proposed a comprehensive efficient flooding algorithm using directional antennas for mobile ad hoc networks to extend our previous work. Our previous proposed algorithm EFDA is based on EF1 that only uses 1-hop neighbor information. However, there are some problems in EFDA, and we attempt to solve these problems in this paper with the target of reducing the number of signal collisions. The results of extensive simulation show that signal collisions can be reduced by using the directional antenna and exchanging the HELLO message. Our algorithm achieves better performance than Pure Flooding, EF1, and EFDA as a result of smaller number of signal collisions. In future, we will study the benefit in other metrics brought by the directional antenna (such as energy consumption), and incorporate our flooding algorithm into ad hoc routing protocols.

References

1. Jiao, X., Wang, X., Zhou, X.: Efficient Flooding for Wireless Ad Hoc Networks with Directional Antennas. In: Proceedings of ISCIT (to be published, 2007)
2. Liu, H., Wan, P., Jia, X., Liu, X., Yao, F.: Efficient Flooding Scheme Based on 1-hop Information in Mobile Ad Hoc Networks. In: Proceedings of IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2006)
3. Park, V.D., Corson, M.S.: A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In: Proceedings of IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (1997)
4. Perkins, C.E., Belding-Royer, E., Das, S.R.: Ad hoc On-Demand Distance Vector (AODV) Routing. In: RFC 3561 (2003), http://www.ietf.org/rfc/rfc3561.txt
5. Johnson, D.B., Maltz, D.A., Hu, Y., Jetcheva, J.G.: The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). In: IETF Internet Draft (2002), http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt
6. Ni, S., Tseng, Y., Chen, Y., Sheu, J.: The broadcast storm problem in a mobile ad hoc network. In: Proc. of ACM/IEEE MOBICOM, IEEE Computer Society Press, Los Alamitos (1999)
7. Cai, Y., Hua, K.A., Phillips, A.: Leveraging 1-hop Neighborhood Knowledge for Efficient Flooding in Wireless Ad Hoc Networks. In: 24th IEEE International Performance Computing and Communications Conference (IPCCC), Phoenix, Arizona (2005)
8. Yang, C.-C., Chen, C.-Y.: A Reachability-Guaranteed Approach for Reducing the Broadcast Storms in MANETs. In: Proceedings of IEEE Semiannual Vehicular Technology Conference, IEEE Computer Society Press, Los Alamitos (2002)
9. Wu, J., Li, H.: On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In: Proc. of the 3rd Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DiaLM) (1999)
10. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: Proc. of ACM MOBIHOC, ACM Press, New York (2002)
11. Spyropoulos, A., Raghavendra, C.S.: Energy Efficient Communications in Ad Hoc Networks Using Directional Antennas. In: Proceedings of IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2002)
12. Choudhury, R.R., Vaidya, N.H.: Performance of ad hoc routing using directional antennas. Journal of Ad Hoc Networks (2005)
13. Tang, J., Xue, G., Chandler, C., Zhang, W.: Interference-Aware Routing in Multihop Wireless Networks using Directional Antennas. In: Proceedings of IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2005)
14. Hou, Y.T., Shi, Y., Sherali, H.D., Wieselthier, J.E.: Online Lifetime-Centric Multicast Routing for Ad Hoc Networks with Directional Antennas. In: Proceedings of IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2005)
15. Orda, A., Yassour, B.-A.: Maximum-Lifetime Routing Algorithms for Networks with Omnidirectional and Directional Antennas. In: Proc. ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC), ACM Press, New York (2005)
16. Wieselthier, J.E., Nguyen, G.D., Ephremides, A.: Energy-aware wireless networking with directional antennas: The case of session-based broadcasting and multicasting. IEEE Transactions on Mobile Computing 1(3), 176–191 (2002)
17. THE VINT PROJECT. The UCB/LBNL/VINT Network Simulator—ns (version 2), http://mash.cs.berkeley.edu/ns

GTCOM: A Network–Based Platform for Hosting On-Demand Desktop Computing

Guangbin Xu[1], Yaoxue Zhang[2], Yuezhi Zhou[3], and Wenyuan Kuang[1]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
{xugb04,kuangwy04}@mails.tsinghua.edu.cn
[2] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
zyx@moe.edu.cn
[3] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
zhouyz@mail.tsinghua.edu.cn

Abstract. The current trend of computing paradigm is heading toward the pervasive computing. Nevertheless, with the rapid advancements in hardware, software, and network, the desktop computing will still prevail in the next decade. In light of this, this paper designed and implemented a platform for hosting desktop computing in a network environment, namely, GTCOM. GTCOM can transparently provide a full functionality as a desktop computer and allow computing services including operating systems to be acquired and utilized on-demand from the server. Since all management tasks are centralized on the server, GTCOM can provide user-needed services through zero-managed clients. We implemented a prototype system based on Godson-2 platform. The experimental results demonstrate only negligible overhead is induced which shows feasibility and effectiveness of our approach.

1 Introduction

Centered on humans' need, computing technology experienced in turn the mainframe computing, the desktop-based network computing, and is now heading toward the pervasive computing. Desktop-based or PC-based computing made a great success in the past decades. In our view, this situation may prevail for another decade. However, while the capability of desktop computing is increasingly enhanced, the complexity of hardware and software and hence its management increases, and so does the total cost of ownership (TCO) of computers, especially to owners such as enterprises. One main difference of pervasive computing is that users can transparently acquire corresponding computing services anytime, anywhere. In other words, computing is autonomic. There is no need for users to care about the locations of computing services or the management of computing services because they acquire them transparently in an on-demand manner. Therefore, how to provide an on-demand service-acquiring environment in which users will not be distracted by things other than their needs, such as

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 535–544, 2007.

efforts of management and maintenance, while preserving the merits of desktop computing is a challenge for system researchers.

Numerous approaches have been designed to provide PC-like functionality while to render management complexity. The typical solutions are the network computer [1], the NetPC [2], X-terminal [3], thin clients [4], and virtual machine monitor (VMM)-based methods. The network computer, typically JavaStation by Sun, cannot support full-featured desktop computing. The thin client technologies are claimed as being able to provide a full-featured desktop with a lower TCO. In this approach, all desktop computing tasks are processed on the central server. This increases the resource requirement of the central server. Some applications, for instances, the media playback application, cannot be supported efficiently. Also, some merits of desktop computing such as users' isolated performance guarantees, privacy, and user-level personalization are lost. The VMM-based methods like VMVare[5], Xen[6], and Disco[7] can provide PC's functionality by running different OSs over virtual machines and are claimed of a better management efficiency by facilitating deployment and migration at the system level. However, it still manipulates hosted OSs as locally-resident resource as a PC, for users, the management complexity is the same. In addition, the vitalization they adopt compromises performance, making some killer applications such as video games can not run fluently.

This paper presents GTCOM, a platform for on-demand desktop computing on a network which allows users to transparently acquire needed computing through one machine. While having full desktop-experienced computing with negligible performance degrading, users can be free from any management effort. GTCOM considerably outperform competing commercial and freely available solutions considering the functionality and performance combined.

The remainder of this paper is structured as follows. In Section 2, we give an overview of the GTCOM architecture and outline how GTCOM works. Section 3 describes the key aspects of our design and implementation. Section 4 addresses the implementation of a prototype based on the Godson-2 platform. Section 5 evaluates the performance of the GTCOM prototype comparison with the desktop computer. Finally, Section 6 concludes.

2 Approach and Overview

The GTCOM system, deriving from the concept of Transparent Computing[8], is constructed with a conventional client/server architecture. It consists of dozens of GTCOM client machines and servers. The client machines and the servers are connected by pervasive networks. The client machines could be various pervasive computing devices, such as common personal computers, but with different amounts of or even without secondary storage. A typical GTCOM system is illustrated in Fig. 1.

No operating system (OS) or application is pre-installed on clients. Instead, they are centrally stored and managed as computing services in servers. Nevertheless, all the user-needed application computing is performed locally on clients rather than on servers. According to the user's choice, the client will automatically fetch the needed services from a server through block-streaming, and then execute on the local

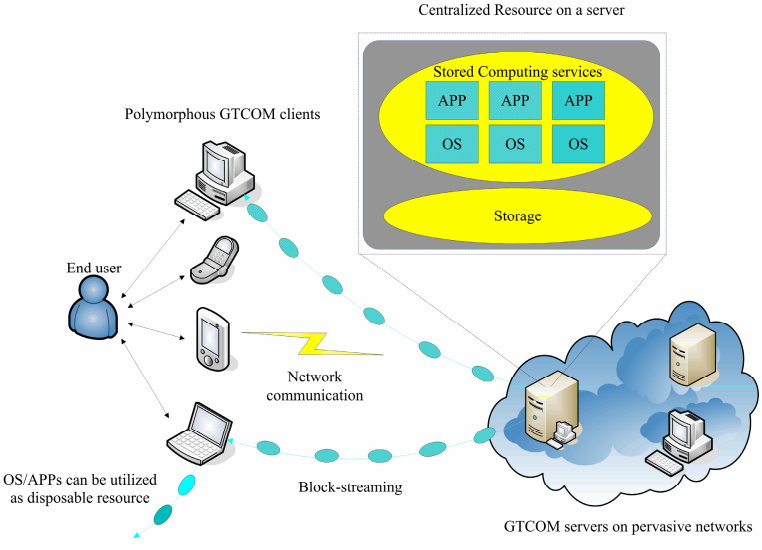


Fig. 1. The Computing Environment of GTCOM

hardware platform. Therefore, the GTCOM system allows programs, even operating systems, to be acquired and be utilized as computing services on demand.

The functionality of GTCOM fall into two stages: boot time and runtime. When powered on, GTCOM clients will boot remotely through a multi-OSs remote booting protocol named the Enhanced Network-based Client Boot Protocol (ENCBP), which can boot different OSs across networks. Another key concept in GTCOM is the VDisks, which are block-based virtual storage devices that actually reside on the server as image files and are accessed through network protocols. By replacing the standard BIOS disk access function, VDisk allows the operating system to boot as on a physical disk. In addition, before entering the runtime period, the OS will first load the network device and then the VDisk driver. The OS then can seamlessly access the VDisk contents through the VDisk driver transparently.

The virtual memory and swapping mechanism is a dynamically memory caching mechanism across a network, which is a key component of GTCOM to enhance runtime performance and enable resource block-streaming. Since the executed applications may need a larger memory than the local physical memory of a client machine, and there is supposed to be no secondary storage device in the client, we therefore extended the paging and schedule technique widely adopted in traditional operating systems to the client/server environment.

3 Detailed Design

3.1 ENCBP: The Multi-OSs Remote-Booting Protocol of GTCOM

In GTCOM, the first issue need to be addressed is how to remote boot an operating system when powered on. We present a multi-OS remote-booting protocol for

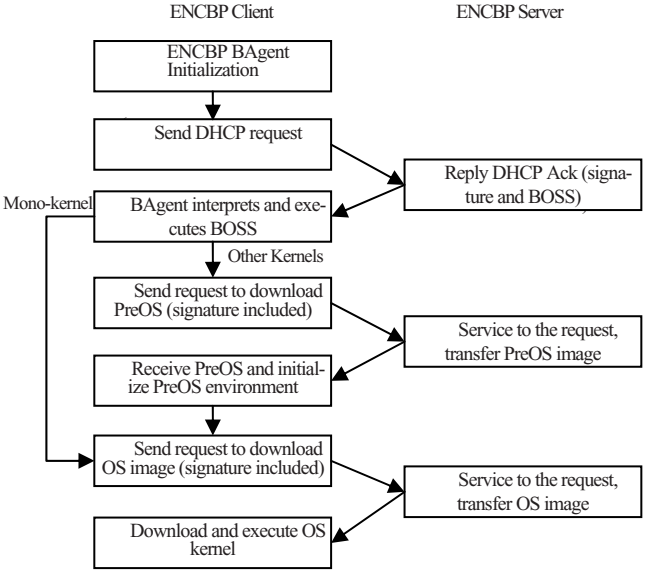


Fig. 2. The process flow of ENCBP

GTCOM—ENCBP, which is based on the client/server model. ENCBP works starting from the client being powered on to the time that machine control power has shifted to the executed OS. In GTCOM, the ENCBP client (EClient) codes are added to the BIOS firmware of the client machine. EClient will take over the control after the POST process, and hence the ENCBP protocol flow starts, as illustrated in Fig. 2.

The instance of EClient codes functions as a booting agent, namely, BAgent, The initialization step includes verifying the validation and integration of EClient codes, and setting up the local executing environment and networking environment. After that, the BAgent obtains network identification by DHCP protocol. There are signatures for subsequent interaction with the server and a booting OS select (BOSS) script encapsulated in the extended DHCP options. BOSS is maintained on the server, and it consists of the description information of available operating systems on the server, including their kernel structure type. The BAgent then executes BOSS by explanation, indicating the user to choose the operating system he/she wants to use. Next, according to the kernel type of the chosen OS, the BAgent will download and boot the kernel image if a mono-kernel OS is chosen. Otherwise (e.g., if it is a hybrid kernel), the BAgent will download and execute an OS pre-execution environment, namely, PreOS. PreOS contains corresponding OS loaders for hybrid kernel-type OSs, such as WINDOWS2000. This is because there is no clear kernel image of them. In this case, the OS kernel image will be downloaded and booted through the PreOS. Then the whole process of ENCBP will end. In order to improve the security and efficiency of the protocol, all the transitions adopt a sector-based transferring protocol which integrates data encryption and signature verification.

3.2 The Block-Streaming-Supporting VDisk

VDisks are virtual block-based devices in GTCOM clients, which are accessed through the IP-based VDisk access protocol (VDAP) network protocol. In other words, it is a network storage mechanism. VDisk provides a linear addressable data block storage access for GTCOM clients, whose contents are actually contained in VDisk images dwelling on the server.

When the file system of a client needs to request some blocks of a VDisk, it will request them in the form of the start block number, the block length, and the operation type (Read/Write/Ack) from the VDisk driver. The VDisk driver will construct a VDAP packet and encapsulate parameter information including the start block number, the block length, the operation type, user ID, disk num, and checksum into the packet. The VDAP packet then will be sent to the server. After this, the VDisk service on the server receives and parses the VDAP packet. The VDisk Service will then verify the users and the VDisk type. After which, the request is conducted using the corresponding VDisk images. Subsequently, the VDisk service constructs a response packet with an operation type of acknowledgement according to the result obtained, and then returns it to the sending clients. After receiving the response packets, the VDisk driver on the GTCOM client will distill resulting blocks from the packets and then hand these to the file system. Fig.3 is the illustrative view of the data access of GTCOM through VDisk.

VDisk Driver is a client program which receives the block requests from the OS and redirects them to the VDisk service on the server through VDAP. The VDisk Driver functions similarly to a real hard disk driver, which receives the block request but then serves the request by reading or writing the blocks stored on the VDisk image on the server, instead of a real hard disk.

The VDisk service takes charge of implementing a disk request to the concrete disk blocks. Consequently, given logical blocks number (LBN) can be flexibly mapped into different disk images or partitioned according to different management strategies.

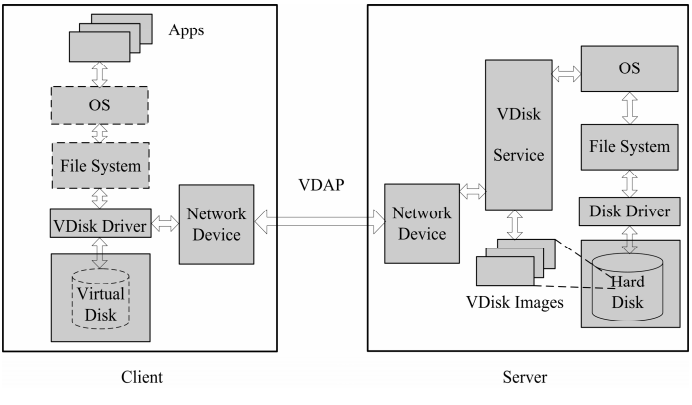


Fig. 3. VDisk-based data access in GTCOM

The VDisk mechanism has several advantages. Its implementation is simple because it just redirects the block level access request, unlike network file systems in which it must accomplish the file system semantics. Moreover, it is independent of a file system layout and also of operating systems. This flexibility and transparency are very important to the on-demand desktop computing of GTCOM. Finally, the VDisk can be flexibly mapped to similar or different disk images by the VDisk service on the server. Thus, it can transparently achieve sharing and privacy to users.

3.3 Virtual Memory and Swapping

Virtual Memory and Swapping (VMS) is a mechanism GTCOM uses to remotely fetch data that are absent in local memory through page-mapping over the network. Since all the programs and data are stored in the server, when the cache miss generated and the needed pages do not exist in the local memory, the GTCOM client needs to dynamically translate and fetch pages remotely from the server. In essence, VMS extended the paging memory management techniques that are widely used in modern operating systems to a client/server environment. Each time a program is executed, the OS creates a new process for it. Meanwhile, a corresponding process space is allocated to retain data and instruction of the process. During the running of the process, whenever a data or instruction page is missed and the interrupt is called, if a page cannot be found in the local memory, VMS inquiry swaps the area in the server through the VDisk Driver. If the page is in it, then it will be fetched directly. Otherwise, it will be fetched from the VDisk image on the server and be sent back to the client.

Through the mechanisms of VMS and VDisk, GTCOM allows GTCOM clients to block-stream program execution and data access during run-time.

4 Implementation

The prototype system of GTCOM was built on Godson-2[9], which is one of China's first-generation CPUs.

As shown in Fig. 4, the prototype consists of five Godson2 clients and three servers. As mentioned, all computing tasks are conducted locally on clients. Servers are not responsible for any applied computing. They are only used for providing software resource that clients will request on-demand. Users can choose to run any available OS after they power on the clients. The OS and application will then be fetched dynamically from the servers. During the whole process, users just transparently use the client as a desktop computer installed with a multiple OSs.

We subdivide the GTCOM server functionality into three in implementation to facilitate implementation and to bring flexibility for tuning performance afterwards. The three functional servers are the network computing access server (NCAS), the OS and applications server (OSS), and the virtual storage server (VSS). NCAS runs the server-side of ENCBP. It is responsible for receiving network access requests, configuring parameters automatically for them, and distributing BOSS script, signature, and PreOS. NCAS also maintains the OS status information for client boot. OSS stores various operating systems and application images for clients' use. After booting, OSS provides read-only VDisk system images. VSS provides virtually extended memory and read/write storage through VMS and VDisk.

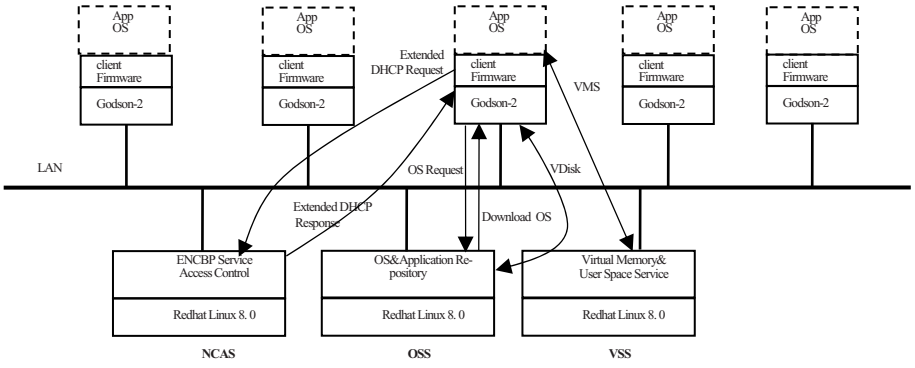


Fig. 4. Architecture of Godson-2 based GTCOM prototype system

ENCBP is developed into two versions: multicast and unicast. The BAgent of ENCBP is designated to download OS images from the OSS. We also classified VDisk into three functional types supporting desktop computing: private disk, sharing disk, and shadow disk, accordingly. The sharing disk is used mainly for storing read-only pristine system data and can be read by all users and GTCOM clients. The private disk stores persistent users' proprietary data. The shadow disk is used for storing historically-modified system data which can support system rollback when needed. The concept of a shadow disk is similar to the ELEPHANT file system in [10]. Their main difference is that we implement on the gratuity of a disk block other than the file gratuity. The private disk, sharing disk, and shadow disk combined can provide a full view of a normal hard disk. The VDisk driver is embodied as an interrupt handler in real mode used to access the VDisk during booting time and a more complex device driver during runtime which inserted into the commodity operating system in which it registers itself as a type of block device in order to work correctly.

5 Evaluation

We conducted a series of experiments according to two different stages of the GTCOM client: boot time and runtime. We began by benchmarking GTCOM's booting efficiency against a real Godson2 PC counterpart, demonstrating that the desktop computing provided by GTCOM has comparable boot performance against a PC, only a negligible delay is induced. Then in order to test the runtime performance of GTCOM, we tested the VDisk throughput against the hard disk of the Godson-2 PC with several experiments using a different memory size and buffer option. These tests demonstrate that we can get the same disk throughput as a normal desktop PC when an optimized hardware configuration of GTCOM is adopted. We then evaluated the functionality performance by conducting a set of experiments in which various application programs were executed on the clients. The results showed that GTCOM can transparently provide the same functionality to end use as a normal PC, in which only negligible overheads are induced.

In all these experiments, the GTCOM prototype consisted of up to five (due to the available amount) Godson2-based clients and three Intel x86-based normal PCs used as three functional servers described in the previous section. All the clients and servers were connected by a 3COM 10/100M LAN switch. The GTCOM clients were constructed with a specialized Godson2 product without a local disk following this hardware configuration: CPU: Godson2 360MHZ; RAM: DDR 133, 256MB (indicated otherwise); network card: INTEL EEPRO100 100Mbps, chipset GT64240B, and BIOS ROM: AMD29F040 32pin. The three servers had the same configuration: CPU: AMD Athlon (TM) 64 Processor 2800+; RAM: Dual DDR 400, 1G MB; Hard Disk: Software RAID0 based on two Seagate Barracuda 7200 RPM; and Onboard network card: Realtek 8139 100Mbps. The counterpart PC was the above GTCOM client plus a local disk (Seagate ST380011A). The operating system of the server was RedHat 8. The operating system of GTCOM client was Debian GNU/LINUX 3.0 for MIPSEL. All the numerical results presented are averaged values from 10 repetitive testing rounds.

We measured time delays for booting GTCOM clients and the counterpart PC for comparison. The booting delay time is defined as the time when the power button has been pushed to the time the XWindow appears. The results are shown in Fig. 5.

As shown in Fig. 5, the GTCOM boot times are comparable to that of the standalone Godson-2 PC, and the booting delay is generally within the acceptable span of users. However, the booting time delay of a GTCOM client using ENCBP in two modes is a little bit longer than that of a PC. That is because the boot process of a GTCOM client involved some overheads of network interactions.

Secondly, we tested the throughputs capability of VDisk and real local hard disks for evaluating GTCOM clients' runtime performance. The throughput capability is defined as throughputs within one second when running the random read/write tasks.

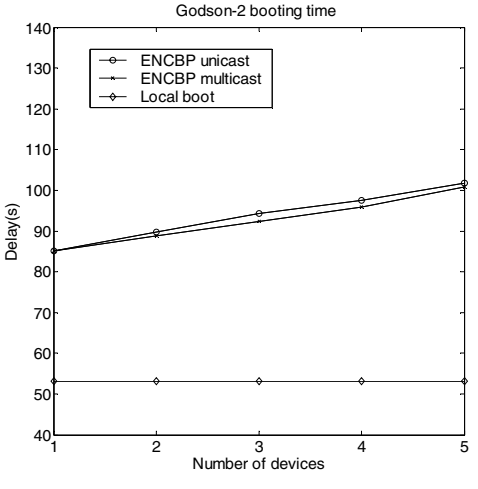


Fig. 5. Booting times of the Godson-2 based GTCOM clients and a Godson-2 PC

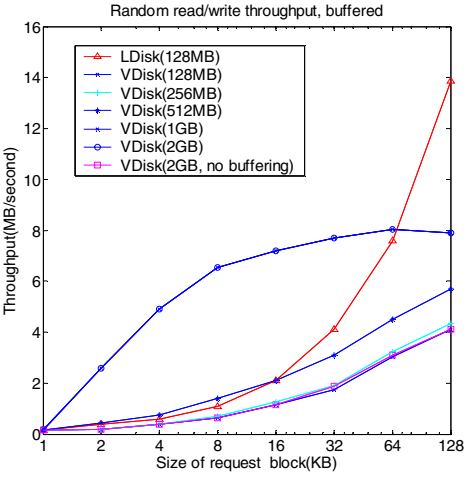


Fig. 6. Throughput comparison of different memory-sizes GTCOM clients with a PC

Table 1. Testing results of GTCOM-hosted desktop computing

Applications/OS	Testing cases description	Delay/ Situation (1 client)	Delay/ Situation (2 clients)	Delay/ Situation (5 clients)
Operating system: Debian LINUX 3. 0	From power-on to the presentation of XWindow	85.01 (s)	89.72 (s)	101.21 (s)
Document process: Openoffice2.0	Startup	29.79 (s)	37.56 (s)	55.63 (s)
	Opening a 1MB MS Word file	4.03 (s)	5.51 (s)	7.87 (s)
	Opening an 18.7MB MS PowerPoint file	23.30 (s)	33.47 (s)	46.89 (s)
WWW: Mozilla Firefox 1.05	Startup	25.41 (s)	32.11 (s)	45.83 (s)
File copy	Copying a 50MB file between two directories	31.71 (s)	37.13 (s)	49.90 (s)
	Copying 20005KB between two directories	13.12 (s)	15.74 (s)	29.91 (s)
Video play:MPlayer	Playing MPG, DAT, BIN, VOB, ASF, WMF, AVI etc.	Fluent	Fluent	Fluent
Audio play: XMMS	Playing mp3, wav, etc.	Fluent	Fluent	Fluent
Instant message: Gaim	Using instant message tool kits includes MSN, ICQ, etc.	Normal	Normal	Normal

The values were measured by an IOMeter configured with default option. We tested according to a different memory size and with/without a buffer on the server. The testing results are shown in Fig. 6. Except for the 2GB memory case which was measured both with and without buffering on the server, all cases are conducted with buffering. As shown, although a PC can have better performance (i.e., throughput) than a GTCOM client in most cases. Nevertheless, when there are small amounts of GTCOM clients running simultaneously and having relatively abundant memory, such as the case when there are four running GTCOM clients with 1GB memory, the GTCOM client can get even more throughput than the Godson2 PC. This is because if we adopt a high-speed network environment such as the 100MB LAN in this

experiment, a sufficient memory for caching, and an appropriate buffering strategy, the disk will become the speed bottleneck when accessing stored data. Therefore，the GTCOM can get a similar performance as a normal desktop PC only if the configuration is reasonable.

In order to evaluate the GTCOM in user's viewpoint, we also executed a collection of the most-used PC application programs. The results are shown in Table 1.

6 Conclusion

In this paper, we proposed a new system, GTCOM, for hosting on-demand desktop computing over networks. While the fully personalized functionality of a PC can be transparently provided, users can be free from management tasks. In GTCOM, all computing services used by users including OSs are acquired over networks on-demand. This is achieved by introducing: the ENCBP to remotely boot multiple OSs from GTCOM clients, the Virtual Disk mechanism and the Virtual Memory and Swapping mechanism to block-stream the services. We implemented a simplified Godson-2 based prototype to evaluate the GTCOM. We also conducted a series of experiments on the prototype. The results showed that GTCOM is an effective approach to host on-demand desktop computing over networks with negligible overhead, and is a feasible paradigm to be applied for providing on-demand computing services toward pervasive computing.

References

1. Comerford, R.: The battle for the desktop. IEEE Spectrum 34(5), 20–28 (1997)
2. Scheifler, R.W., Gettys, J.: X Windows System, 3rd edn. Digital Press (1992)
3. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual Network Computing. IEEE International Computing 2(1), 33–38 (1998)
4. Schmidt, B.K., Lam, M.S., Northcutt, J.D.: The interactive performance of SLIM: a stateless, thin-client architecture. In: 17th ACM Symposium on Operating System Principles (SOSP), Kiawah Island Resort, SC, vol. 34, pp. 32–47 (December 1999)
5. Devine, S., Bugnion, E., Rosenblum, M.: Virtualization system including a virtual machine monitor for a computer with a segmented architecture. US Patent, 6397242, Oct (1998)
6. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T.: Xen and the Art of Virtualization. In: SOSP 2003 (October 2003)
7. Bugnion, E., Devine, S., Govil, K., Rosenblum, M.: Disco: Running commodity operating systems on scalable multiprocessors. In: Proceedings of the 16th ACM SIGOPS Symposium on Operating Systems Principles. ACM Operating Systems Review, vol. 31(5), pp. 143–156. ACM Press, New York (October 1997)
8. Zhang, Y.: Transparence Computing: concept, architecture and implementation. Chinese Journal of Electronics 32(12)A (December 2004)
9. Hu, W., Zhang, F., Li, Z.: Design and Performance Analysis of the Godson22 Processor. Journal of Computer Research and Development 43(6), 959–966 (2006)
10. Santry, D., Feeley, M., Hutchinson, N., Veitch, A., Carton, R., Ofir, J.: Deciding when to forget in the elephant file system. In: SOSP 1999. Proceedings t. of the 17th ACM Symposium on Operating Systems Principles, Kiawah Island, South Carolina, South Carolina (December 1999)

Multi-robot Task Allocation Using Compound Emotion Algorithm

Wei Yuan[1] and Bi Zeng[2]

[1] Faculty of Computer,Guangdong University of Technology
Guangzhou, China
yuan619wei@126.com
[2] Faculty of Computer,Guangdong University of Technology
Guangzhou, China
z9215@163.com

Abstract. A new approach is proposed in this thesis based on Compound Emotion Algorithm--CEA for the multi-robot task allocation. By comparing both the Greedy Algorithm and Random Algorithm in the average amount of communications among robots, the average amount of communications between robots with certain rate of messages got lost, and the total waiting time for the "HELP" message senders for response, The simulated results show that the Compound Emotion Algorithm works well.

1 Introduction

Mobile robots are applied to wide range of areas such as military, urban search and rescue. As the application increased in more and more complicated environment, The capabilities of a single robot is very limited, the cooperation among a group of robots to extend the application. Therefore, it is important to study how tasks allocated dynamically among robots ,and to avoid the rapid increase of bandwidth because of the increase of communications. so that large increases in the number of robots does not translate to a large increase in required bandwidth. And certain rate of messages got lost in the transmissin should be considered. This approach requires less communication bandwidth than other methods, enabling it to scale to large team sizes, and making it appropriate for low-power or stealth applications.

2 Algorithms Analysis

2.1 Existing Algorithms

For a group of robots to effectively perform a given system-level task, the designer must address the question of which robot should do which task and when. The process of assigning individual robots to sub-tasks of a given system-level task is called task allocation, and it is a key functionality required of any MRS. Dynamic task allocation is a class of task allocation in which the assignment of robots to sub-tasks is a dynamic process and may need to be continuously adjusted in response to changes in the

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 545–550, 2007.

task environment or group performance.There are many multi-robot task allocation algorithms, they emphase on different points. Greedy and Random are the typical Algorithms which focus on one robot sends out “HELP” message and others take response, and both of them are based on contract net protocol. For the Greedy Algorithm ,the basic principle is: when one robot can’t achieve it’s task, then it will send a “HELP” message, all the idle robots take the response immediately, and the message sender will transfer the task to the responser whose fitness is the best. Random Algorithm has the same principle with the former, the little difference is that the Random Algorithm selects responsers randomly without considering their fitnesses. According to the pinciple, Greedy Algorithm requires more bandwidth especially the robot team is large. There are many fitness computing mothed. but people usually use the time to the message sender indicates the fitness. In the Random Algorithms the message sender does not send the task to the best-fitness responser, so it needs more time to wait for the responser’s arrival.

2.2 Compound Emotion Algorithm

The dynamic task allocation scenario we study considers a world populated with tasks and robots that are equally capable of performing each task but can only be assigned to one at any given time. For example, the tasks could be targets of different priority that have to be tracked, different types of explosives that need to be located, etc. Additionally, a robot can be idle--each robot is always performing a task or searching for a task at any given time. The purpose of task allocation is to assign robots to tasks in a way that will enhance the performance of the system, which typically means reducing the overall bandwidth and the average amount of communications. Here, we suppose every robot’emotion has two emotion genes the “anger’ and the “gratitude”, the “anger” gene means the robot gets anger when the message sender broadcasts the “HELP”message frequently or the robot does not get the task. the “gratitude ” gene means the message sender thanks for the response’s help who finally gets the task. Every robot’s emotion is affected by the two emotion genes. Every time when the message sender broadcasts “HELP” message, the robots whose emotions are below a certain threshold can have the right to response, in this way the low-emotion robot will have the opportunity to get “gratitude”and upgrade it’s emotion. but the one whose emotion is above a certain threshold can't response, and it only can minus a "anger" gene from its emotion, this means to reduce its emotion, if its emotion below a certain threshold it can reponse again.

The protocol begins when the requester robot broadcasts a HELP message with its location and a percept that a robot must have to be a responser, and end with other robot gets the task finally. Here, we use broadcast mothed for the communition which can deeply reduce the bandwidth. Therefore, the Algorthim is based on a 2-way TCP/IP handshake, the contents are detailed in Figure1.

The Compound Emotion Agorithm is also based on contract net protocol. the notation used is as follows. Given a team of n robots, $\{r_1,...,r_n\}$,each robot r_i in the team maintains a level of EMOTION m_i , such that $0 \le m_i \le 1$. At the time point t , one robot r_i broadcasts the “HELP” message, all the idol robot would get the

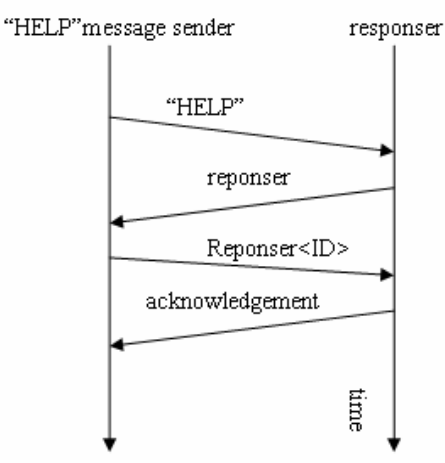


Fig. 1. The communication between message sender and responser

message. here we suppose r_k is the idol robot. each robot r_k will first account for their EMOTION m_k, if r_k robot's EMOTION m_k is lower than a certain threshold Γ, then the robot r_k send the response. but, if r_k robot's EMOTION m_k is not lower than a certain threshold Γ, the "anger" gene will be triggered, this make the r_k robot's EMOTION m_k becomes as below:

$$m_k = m_k - anger * D_k / Dideal \tag{1}$$

here, anger is the "anger " gene. D_k is the fitness of robot r_k, here we also use the time to the message sender indicates the fitness , $Dideal$ is the ideal fitness. thus, the messager sender receives the responses, it will select the best-fitness robot, and transfer the task to responser, then the "gratitude" gene will be triggered and also send to the best-fitness robot who finally gets the task. and this will make the robot's EMOTION becomes as below:

$$m_k = m_k + gratitude \tag{2}$$

"gratitude" gene here is relative to the responsers' number. According the Function (1), we can see that, when the robot gets anger, $anger * D_k / Dideal$ will be subtract from the EMOTION m_i. thus, the EMOTION of the robot drops quickly whose postion is near the messager sender, and the "HELP" message sender can easily transfer its task to the near robot than to the remote one. otherwise, bandwidth will become low when only allowing the robot whose EMOTION is lower than a certain threshold Γ to take response.

3 Simulated Results

In this part ,we make a comparison of the three algorithms using simulation software. Every results is tested 100 times. there are three aspects to be tested:

First, the comparison of average amount of communications between robots with different robot numbers. the numbers of robots are separately 5, 10, 15, 20 and 30, which we assume that each robot to search task, a 50% possibility is unable to complete the task and need to abandon. the contents are detailed in Figure 2 and Table 1. with the results, Random Algorithms grows quickly than others.

Sencond, the comparison of total waiting time for the "HELP" senders for response. the numbers of robots are separately 5, 10, 15, 20 and 30. the contents are detailed in Figure 3 and Table 2. In the Random Algorithm messager senders don't transfer the task to the best-fitess responsers, so they possibily need a much longer

Table 1. Average amount of communications between robots

Robot number / Algorithms	5	10	15	20	30
Greedy	6.75	20	40.5	70.85	143
Compound Emotion	5.5	19.6	38	64.35	123
Random	8.5	21.6	42	72.15	146

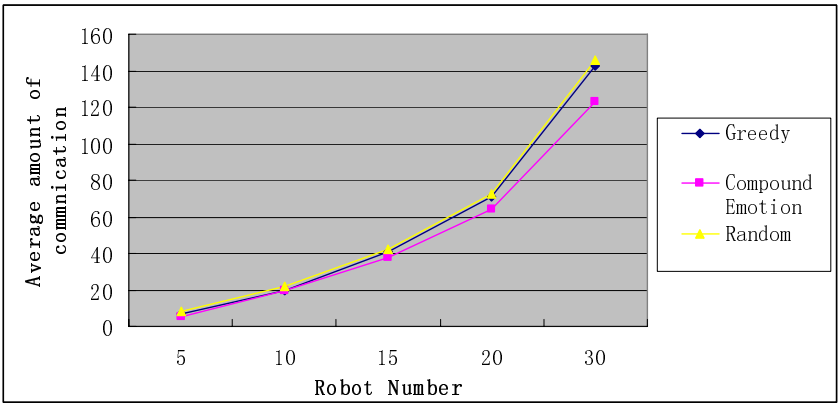


Fig. 2. Average amount of communications between robots

Table 2. Total waiting time for the "HELP" senders for response

Robot number / Algorithms	5	10	15	20	30
Greedy	53.5	75.75	118.85	149.3	179.35
Compound Emotion	59.7	82.1	125.8	154.55	186.65
Random	63.55	104.1	161.2	230.3	296.95

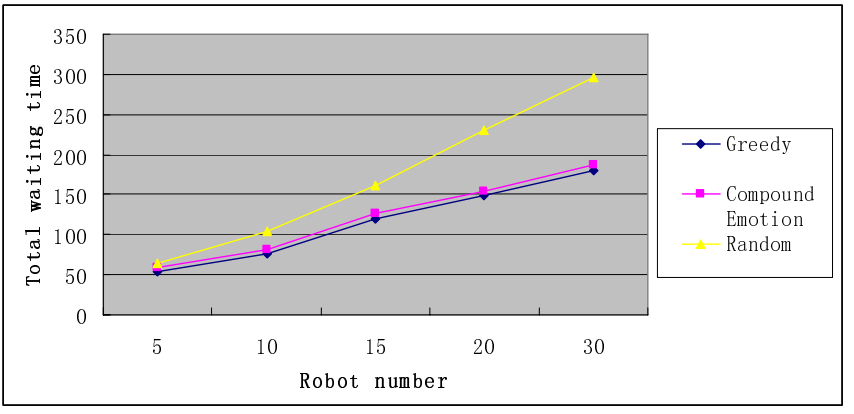


Fig. 3. Total waiting time for the "HELP" senders for response

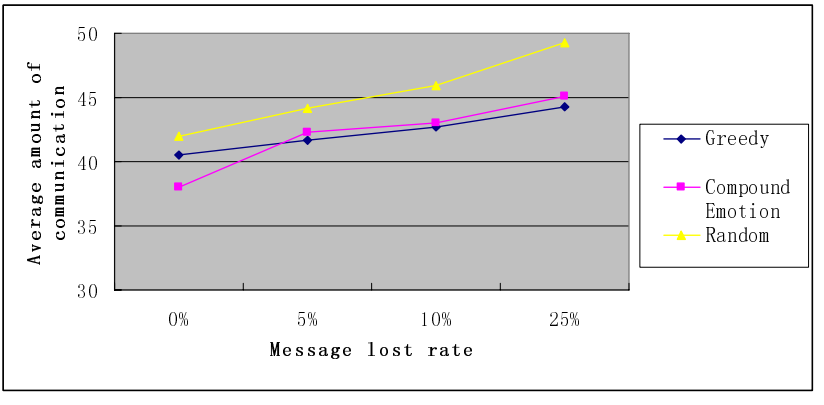


Fig. 4. The comparison of the average amount of communications between robots with certain rate of messages got lost

Table 3. The comparison of the average amount of communications between robots with certain rate of messages got lost

Robot number / Algorithms	0%	5%	10%	25%
Greedy	40.5	41.65	42.75	44.24
Compound Emotion	38	42.25	43.07	45.12
Random	42	44.15	45.93	49.31

time to wait for the responsers'arrival. and Greey Algorithm performs better, because it transfers the task to the best-fitness responser and no threshold .

Third, the comparison of the average amount of communications between robots with certain rate of messages got lost. the lost rate are separately 0%, 5%, 10% and

25%. the robot number is 15. the contents are detailed in Figure 4 and Table 3. Random Algorithm also grows quickly than others, because any time when the requester robot broadcasts a HELP message with its location all the idle robots can take response without considering any threshold. Random Algorithm works worse because it doesn't consider fitness and Compound Emotion Algorithm works between them.

4 Conclusion

In this paper we introduce a new method for the multi-robot allocation, and made a comparison with other algorithms in the simulating environment. with its property, Compound Emotion Algorithm appropriate for low-power or stealth applications. In fact we just tested it in the simulating environment, not in the real robots, that is to say we did not consider the weather conditions and road pavement situations etc in the real enviroment which would affect the robots'performances. otherwise all the robots here are homogeneity, this heterogeneity was not tested in simulation. so we need to study these problems in our future work.

Acknowledgment

The authors are grateful to Guangdong Provincial Natural Science Foundation of China (05001801) for decisive support.

References

[1] Gerkey, B.P., Mataric, M.J.: Sold!: Auction methods for multirobot coordination. IEEE Transactions on Robotics and Automation 18(5), 758–768 (2002)
[2] Parker, L.E.: Alliance: An architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation 14(2), 220–240 (1998)
[3] Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers 29(12), 1104–1113 (1980)
[4] Ortony, A.: On making believable emotional agents believable. In: Trappl, R., Petta, P., Payr, S. (eds.) Emotions in Humans and Artifacts, pp. 189–211, ch. 6. The MIT Press, Cambridge (2002)
[5] Ortony, A.: Subjective importance and computational models of emotions. In: Hamilton, V., Bower, G.H., Frijda, N.H. (eds.) Cognitive Perspectives on Emotion and Motivation, pp. 321–343, ch. 13. Kluwer Academic Publishers, Dordrecht (1988)
[6] Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge (1988)
[7] Gerkey, B.P.: On Multi-Robot Task Allocation. PhD thesis, University of Southern California (August 2003)
[8] Parker, L.E.: Alliance: An architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation 14(2), 220–240 (1998)
[9] Nadig, D., Iyengar, S.S., Jayasimha, D.N.: A new architecture for distributed sensor integration. Proceedings IEEE Southeastcon 1993, pages 8 (April 1993)

The Security Threats and Corresponding Measures to Distributed Storage Systems*

Lanxiang Chen, Dan Feng, and Liang Ming

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan National Laboratory for Optoelectronics, Wuhan, 430074, China
dfeng@hust.edu.cn, lxiangchen@gmail.com

Abstract. There are various threats in distributed storage systems, but there is no comprehensive category. There are some research works on threat modeling and the challenges of protecting storage systems, but there is no corresponding security measure to these threats and challenges. In this paper, elements of storage security and a comprehensive category of threats are presented, which goes from hardware level to software level, from kernel level to application level, from local to network. The corresponding security measures to these threats are provided. And the seven steps process of security is proposed which gives methodology to implement these security measures. Finally, some important storage security issues and future directions to storage security are concluded.

Keywords: storage security, security threat, security measure, process of security, threat modeling.

1 Introduction

The 2006 11th CSI/FBI survey [1] indicates that virus attacks, unauthorized access, laptops (or mobile hardware) and theft of proprietary information (i.e., intellectual property), these four categories account for more than 74 percent of financial losses. New security threats are emerging as storage is increasingly distributed across wide area networks.

Prior to claiming the storage security, it is important to identify the threats. Enumerating the threats helps system architects set up corresponding security measures. There are various threats [2-7], but there is no comprehensive category. Storage Network Industry Association (SNIA) has subdivided storage security into Storage System Security (SSS), Storage Resource Management (SRM), Data In-Flight (DIF), Data At-Rest (DAR) [5]. The 451 Group [8] has identified the threats as theft of privileged access, accidental changes, privileged access abuse, data tampering, application tampering, and hardware theft from the perspective of

* This work was supported by the National Basic Research Program of China (973 Program) under Grant No.2004CB318201, the Program for New Century Excellent Talents in University NCET-04-0693 and NCET-06-0650, Wuhan Project 20061002031 and 200750730307, and the National Science Foundation of China No.60503059 and No.60603048.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 551–559, 2007.

enterprises. Swiderski et al. [9] classify threats into spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege based on their effect. Gruener et al. [7] give an incomplete subset of possible threats and their locations. All these categories are incomplete. For completeness, we categorize security threats from hardware level to software level, from kernel level to application level, from local to network.

In addition, there are some research works on threat modeling [2-4, 9] and the challenges of protecting storage systems [2, 7]. Reference [2] gives two comprehensive processes to design storage protection solutions which are based on classical security principles (Confidentiality, Integrity, Availability, Authentication, or CIAA) and the data lifecycle model. Myagmar also gives some threat modeling approaches [3, 4]. But they don't give corresponding security measures to these threats and challenges.

In this paper, elements of storage security and a comprehensive category of threats are presented, which goes from hardware level to software level, from kernel level to application level, from local to network. The corresponding security measures to these threats are provided. And the seven steps process of security is proposed which gives methodology to implement these security measures. Finally, we conclude some important storage security issues and future directions to storage security.

The remainder of this paper is organized as follows. Section 2 discusses the security threats of distributed storage systems and corresponding security measures. Section 3 discusses some important storage security issues. Future directions to storage security are discussed in Section 4.

2 The Security Threats of Distributed Storage Systems and Corresponding Measures

In this section, we will discuss the security threats of distributed storage systems and corresponding security measures. Elements of storage security and a comprehensive category of threats are presented. The corresponding security measures to these threats are provided. And the seven steps process of security is proposed which gives methodology to implement these security measures.

2.1 Elements of Storage Security and Security Threats of Storage Systems

In contrary to SNIA [5], who has subdivided storage security into SSS, SRM, DIF and DAR described above, we subdivide storage security into the security of storage devices, storage systems, storage network, and storage application. They make up of the whole storage security as Figure 1. There are various threats in any element of storage security.

When designing a storage protection solution, the security engineer should weigh the value of each security measure versus the threats presented in the specific environment. It is important to understand all the threats presented in a storage system before designing any storage protection solution because the threats determine the corresponding security measures.

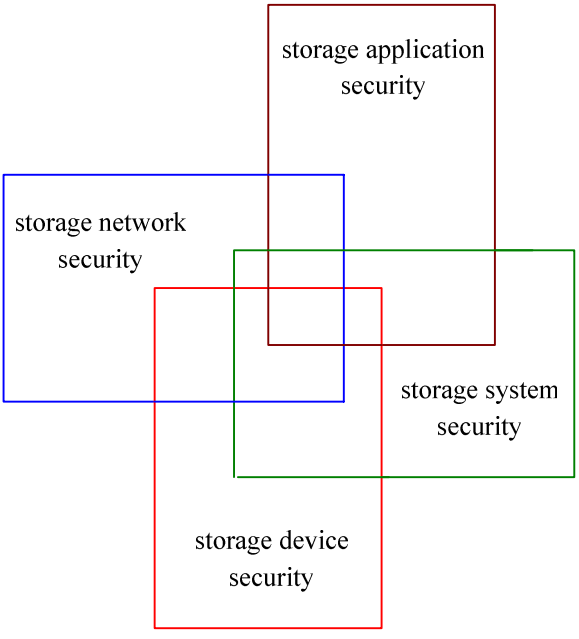


Fig. 1. Elements of storage security

For completeness, we categorize security threats from hardware level to software level, from kernel level to application level, from local to network. They are threats that come from storage devices, storage systems, storage network, and storage application. They are illustrated as Figure 2.

2.2 Corresponding Security Measures

The earliest security measure is just simple authentication, such as user/password. Since the earliest network is not as pervasive and advanced as this moment and attack means are little. Along with the development of network, besides conventional cryptography and access control techniques, there are a series of security mechanisms and techniques, such as anti-virus, firewall, credential-based authentication and authorization, redundancy, backup, versioning, immutable, tamper-proof, intrusion detect and protection, logging, audit and accounting etc.

The core techniques of storage security must go from storage devices and storage system, through storage network, to storage application. The overall security of a distributed storage system is a system project: a system is only as secure as its weakest link, any link will be the attack point, and it is going to be attacked at its weakest point.

The security of storage devices refers to availability and reliability of storage devices. The availability refers to that the system would facilitate the restoration of fault-tolerant, and take out of the demotion state when devices fail down. It is assessed by the Mean-Time-To-Repair (MTTR). The reliability refers to that the system would still complete data storage task in the demotion state when devices fail

down, through Mean-Time-To-Failure (MTTF) to assess. For threats coming from storage devices, it can be evaded through redundancy, backup, versioning etc. Redundancy such as RAID [10] can improve reliability of storage devices. Remote backup [11] can avoid risks coming from disaster, terrorist etc. If some partitions of storage devices fail down, then versioning [12] can help roll back to the former version.

The security of storage systems refers to the security of system software, such as operating system, file system and application software etc. The security measures are access control, authentication and authorization, anti-virus software, firewall and intrusion detection and protection etc.

The security of storage network ensures the security of storage network, through cryptography, access control, authentication and authorization, intrusion detect and protection etc.

The security of storage application ensures the logic security of data stored in storage devices, through cryptography, access control, authentication and authorization, immutable, tamper-proof, logging, audit and accounting etc.

To different security threats, the corresponding security measures are summarized as Table 1.

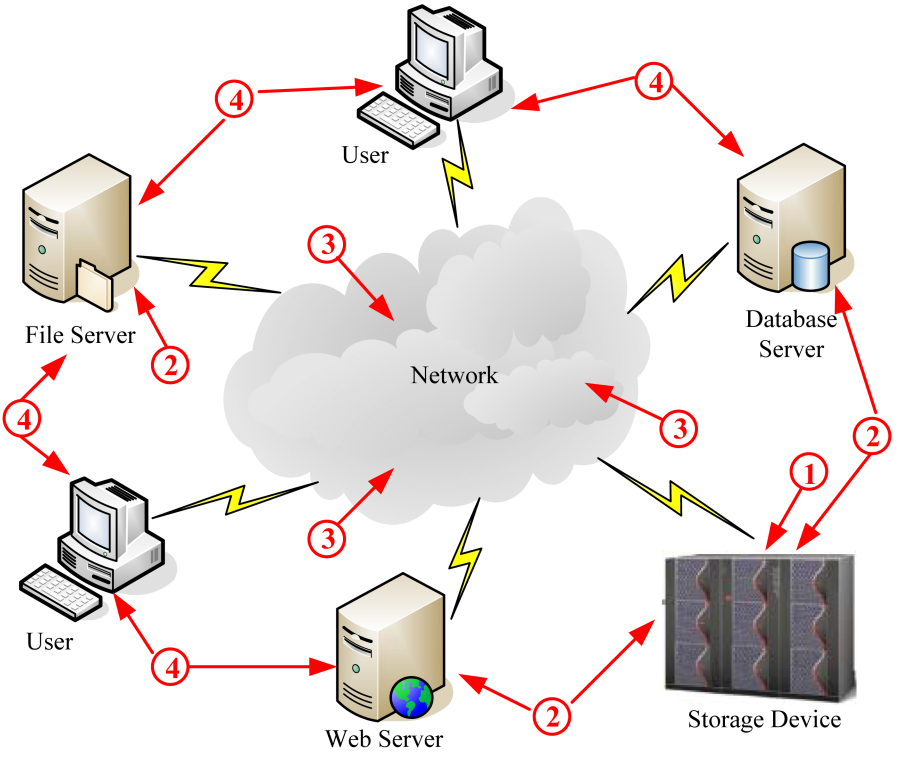


Fig. 2. The security threats of distributed storage systems. 1. The threat comes from storage devices; 2. The threat comes from storage systems; 3. The threat comes from storage network; 4. The threat comes from storage application.

Table 1. Security threats and corresponding measures

Security threats come from	Corresponding security measures
Storage devices	Redundancy, backup, versioning
Storage systems	Access control, authentication and authorization, anti-virus, firewall, intrusion detection and protection
Storage network	Cryptography, access control, authentication and authorization, intrusion detect and protection
Storage application	Cryptography, access control, authentication and authorization, immutable, tamper-proof, logging, audit and accounting

2.3 Methodology - The Seven Steps Process of Security

The seven steps process of security is proposed to implement security measures. Besides CIAA, there are authorization, redundancy/ backup/versioning, audit/accounting, and intrusion detect and protection. Figure 3 illustrates these steps. It is described as follows.

Step 1: authentication to verify whether you are what you claim.
Step 2: authorization to tell you what you are permitted to do.
Step 3: audit/accounting to audit what you have done.
Step 4: integrity ensures the correctness of the data and prevents unauthorized modification of data. It will be detected when data is modified.
Step 5: confidentiality ensures that only those who hold the proper key to access to data. Even data has been compromised, adversaries can not get any valuable information.
Step 6: redundancy/backup/versioning ensures data availability uninterruptedly when disaster, terrorist etc. occurs.
Step 7: intrusion detect/immutable/tamper-proof defends in advance to prevent attack and vicious tampering.

Integrity can be implemented through cryptographic hashes, keyed hashes and public key signature etc. Confidentiality can be implemented through cryptographic algorithms, such as DES [13] and AES [14] etc. Along with the seven steps process of security, it should provide corresponding key management and sharing, including key generation, distribution, storing, recovery and revocation.

It is important to note that it will invite disaster in that if the chosen storage protection solution does not match the threats and vulnerabilities in the actual system resulting in wasted investment, performance degradation, data compromise, service denial, or worse. So it may not need all these seven steps security measures, the security engineer should tailor each security measure according to the specific environment.

After the previous steps have been completed, it is time to think about management and administration interfaces. There are many inside threats coming from authorized users, such as mistake or purposive operations. First, the storage system must be designed sophisticatedly to prevent intended attacks from insider. Then managers and administrators should take part in some special technical training.

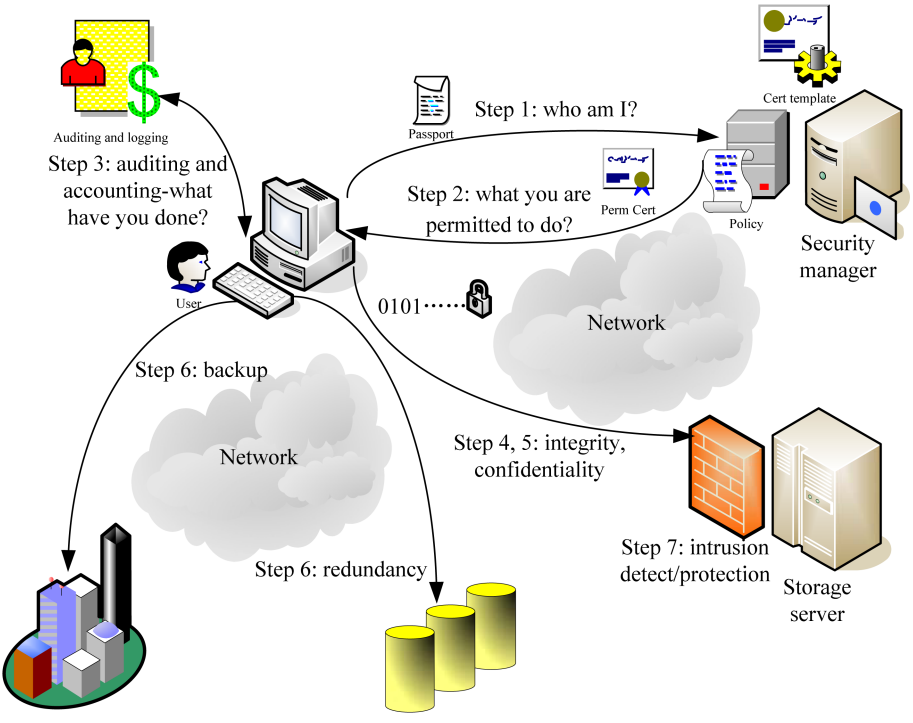


Fig. 3. The seven steps process of security

"Today, storage infrastructures (disk, arrays, IP and SAN fabrics, NAS and tape) are highly vulnerable to attack because of the gap between known security techniques and their level of implementation." Hibbard et al. said [5]. Exactly, it is inadequate to have these security measures. The most important thing is to implement them sophisticatedly.

3 Some Important Storage Security Issues

The main goals of distributed storage system are high scalability, security, while maintaining simplicity and performance, and keeping the cost of implementation low. To achieve these goals, there are some important issues of storage security.

3.1 Appropriate Security Model

Security model determines trusted components, key paths, threat points of systems. And all these will determine security measures. Conventional security model is based on two parties. One party is on both control path and data path which tends to be bottleneck. Object based store system (OBS) [15-17] is based on three parties, which separates control path from data path and puts most meta-data management to object

store device (OSD) to provide scalability. However, the tri-party security model needs more trusted components and results in more threat points.

Appropriate security model includes appropriate trusted component, appropriate control path and data path, appropriate security measures and policies in appropriate threat point.

3.2 Comprehensive Security Measures

To secure a distributed storage system, besides conventional cryptography and access control techniques, there are a series of security mechanisms and techniques, such as redundancy, backup, versioning, immutable, tamper-proof, intrusion detect and protection, logging, audit and accounting etc. A system is only as secure as its weakest link, any link will be the attack point, and it is going to be attacked at its weakest point. So it is important to set up comprehensive security measures. On the other hand, it will invite disaster in that if the chosen storage protection solution does not match the threats and vulnerabilities in the actual system, it also results in wasted investment, performance degradation, data compromise, service denial, or worse. It should carry out corresponding measures according to specific application environment.

3.3 High Performance Implementation

Any security measure will reduce performance, cryptography, access control, immutable, tamper-proof, intrusion detect and protection, and audit will take certain time and space cost, while redundancy, backup, versioning and logging will take certain space cost. High performance implementation refers to reduce the impact of security measures on performance and keep the cost of implementation low.

In addition, efficient access control techniques, efficient encryption and integrity schemes, efficient key management schemes, and transparent to user are also the important issues of storage security.

4 Future Directions to Storage Security

High scalability, high performance and security are the requirements of distributed storage systems. Along with more complex network and larger scale storage system, hybrid security measures, intelligent and self-organization security, and the standardization of storage security are inevitable directions.

4.1 Hybrid Security Measures

Respond to complex environment and various attack means, one security measure is inadequate to ensure security. It needs hybrid security measures to cope with different threats. As detailed above, the security engineer should think about the specific threats to their system first, then set up corresponding security measures.

4.2 Intelligent and Self-organization Security

Smart storage needs intelligent security. Intelligent and self-organization security refers to that storage systems themselves organize security measures according to specific application environment and organize defense and recovery mechanisms according to specific threat without human intervention and management as possible or practical.

Future smart storage devices utilize themselves' computing capability to provide more semantic support. How to utilize the intelligence of devices to improve the self-organization capability of storage systems is also the future direction.

4.3 The Standardization of Storage Security

When deciding which security measures are provided for a storage system, the security engineer should weigh the value of each security measure versus the threats and vulnerabilities presented in the specific environment. Engineers should consult storage security standards since they may provide useful policies. These standards can improve the quality and interoperability, and provide broader heterogeneous approaches of various storage systems. The standardization of storage security has made its debut in recent years.

The combination of iSCSI and Fibre Channel could bring a security threat if handled poorly by customers and storage networking vendors. The storage network security standard - Fibre Channel Security Protocol (FC-SP) [18] addresses many of these concerns, and is backed by many storage networking vendors. FC-SP includes protocols to authenticate and establish secrets for Fibre Channel entities, protocols for frame-by-frame integrity and confidentiality, and protocols to define and distribute security policies within the fabric. The SNIA's Storage Management Initiative Specification (SMI-S) [19] provides a common approach to manage devices in a storage network, using the common information model (CIM) as a foundation and SSL for secure management. Both standards facilitate storage protection by building consensus between vendors to allow interoperability and broader heterogeneous approaches. IEEE [20] proposes a draft P1619 which describes a tweakable wide-block encryption for disk sector level.

Some protectors advocate non-standard storage implementations based on security-by-obscurity [21]. Since a widely implemented standard must be solid since any vulnerability in a standard implementation would be very attractive to attackers to use as part of a class-break exploit (standard becomes a threat). We do not think so. As standards will help the storage security realm build consensus between vendors, which will allow interoperability and broader heterogeneous approaches. There is an analogy between the standardization of storage security and the standardization of cryptographic algorithms, it provides interoperability while convinces users of their security. If I don't know what you have done for security, how can I believe that it is secure?

References

1. Gordon, L.A., Loeb, M.P., Lucyshyn, W., Richardson, R.: Computer Crime and Security Survey. Compute Security Institute (2006)
2. Hasan, R., Myagmar, S., Lee, A.J., Yurcik, W.: Toward a Threat Model for Storage Systems. StorageSS (2005)

3. Myagmar, S., Lee, A.J., Yurcik, W.: Threat Modeling as a Basis for Security Requirements (SREIS). In: Symposium on Requirements Engineering for Information Security (2005)
4. Myagmar, S.: Threat Modeling Networked and Data-Centric Systems. University of Illinois at Urbana-Champaign, Department of Computer Science M.S. Thesis, August (2005)
5. Hibbard, E.A., Budnik, L., Austin, R.: Introduction to Storage Security A SNIA Security White Paper. White Paper (October 14, 2005)
6. Edmonds, A.: Towards Securing Information End-to-End: Networked Storage Security Update and Best Practices. White Paper (February 2003)
7. Gruener, J., Kovar, M.: The Emerging Storage Security Challenge. Yankee Group Report (September 2003)
8. The 451 Group, Storage Security Market: Emerging Opportunities, Unseen Threats. (May 2003)
9. Swiderski, F., Snyder, W.: Threat Modeling. Microsoft Press (2004)
10. Chen, P.M., Lee, E.K., Gibson, G.A., Katz, R.H., Patterson, D.A.: RAID: High-performance, reliable secondary storage. ACM Computing Surveys 26(2), 145–185 (1994)
11. Chervenak, A., Vellanki, V., Kurmas, Z.: Protecting file systems: A survey of backup techniques (1998)
12. Zhu, N., Chiueh, T.: Design, implementation, and evaluation of repairable file service. In: Proceedings of Dependable Systems and Networks (DSN) (2003)
13. National Institute of Standards and Technology, Data Encryption Standard (DES), FIPS Publication 46-3 (October 1999)
14. National Institute of Standards and Technology (NIST).Federal Information Processing Standards Publication 197 (FIPS PUB 197): Specification for the Advanced Encryption Standard (AES) (November 2001)
15. Information technology - SCSI Object-Based Storage Device Commands -2 (OSD-2). T10 Working Draft, (October 2004), http://www.t10.org/ftp/t10/drafts/osd2/osd2r00.pdf
16. Du, D., He, D., Hong, C., Jeong, J., Kher, V., Kim, Y., Lu, Y., Raghuveer, A., Sharafkandi, S.: Experiences in Building an Object-Based Storage System based on the OSD T-10 Standard. In: MSST 2006. Proceedings of the 23nd IEEE / 14th NASA Goddard Conference on Mass Storage Systems and Technologies, College Park, MD (May 2006)
17. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C., Ceph, A.: scalable, high-performance distributed file system. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI). Seattle, WA (November 2006)
18. Information technology - Fibre Channel Security Protocol. T10 Working Draft (February 2006), http://www.t11.org/ftp/t11/pub/fc/sp/06-157v0.pdf
19. Storage Network Industry Association (SNIA) - SNIA Storage Management Initiative Specification (SMI-S) (December 23, 2003)
20. IEEE, Draft proposal for tweakable wide-block encryption (May 2007), http://ieee-p1619.wetpaint.com/?t=anon
21. Chirillo, J., Blaul, S.: Storage Security: Protecting, SANs, NAS and DAS. Wiley (2002)

Research on Dynamic Load Balancing Algorithms for Parallel Transportation Simulations

Dongliang Zhang[1,2], Changjun Jiang[1,2], and Shu Li[1,2]

[1] Department of Computer Science and Technology, Tongji University, Shanghai, China, 201804
[2] The Key Laboratory of "Embedded System and Service Computing", Ministry of Education, China, Shanghai 201804

Abstract. To the issue of dynamic load balancing in parallel transportation simulations, we describe two algorithms for different types of task partitions, parallel lines partition and grid partition. In the algorithms, load balance is obtained by iteratively moving the boundary lines according to the relative balance of adjacent sub-domains. Assuming real traffic distribution as the experimental work load, we test the performance of the algorithms. And the result we observe confirms the value of the methods. Based on the discussion of the communication overheads under different types of partitions, due to the relative small amount of boundary lines, grid partition can decrease the communication overheads and is a more adaptive partition model.

Keywords: load balance, parallel computing, transportation simulation.

1 Introduction

As an important component of Intelligent Transportation Systems (ITS), the large-scale transportation simulation system is widely used in researching and designing transportation control systems. However, a large scale transportation simulation needs a large amount of calculation. The development of distributed and parallel computing technology provides sufficient calculating resource for traffic simulation. In recent years, the research on parallel traffic simulation has gained much progress [1, 2, 3].

In parallel computing, load balancing is an effective way to minimize the processing time and maximize the utilization of calculation resource [5, 6]. In a distributed and parallel transportation simulation system, the work loads of processors are often unbalanced, and so the efficiency of calculation is quite low. The main task of a transportation simulation is to simulate the behavior of vehicles on the road network. And in a view of large scale, the simulation of transportation is much similar to that of molecular dynamics [4]. We can take vehicles as molecules with special moving rules in a two dimensional square. The typical approach to parallelizing these computations is to decompose the spatial region into sub-domains, and associate each sub-domain with a processor. The computational work for a given processor at each time step depends on the number of vehicles in the corresponding spatial domain. Due to the ununiformity of the traffic load in the real word, the simulation tasks differ between

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 560–568, 2007.

sub-domains. In the simulation, when a vehicle is running across the boundary of two sub-domains, the source processor needs to pass it to the destination processor. So the transfer of vehicles changes the distribution of simulation work loads during the simulation procedure. To balance the loads between processors, the boundary lines need to be relocated.

In this paper, we partition the traffic map using parallel lines and quadrilateral grids separately, and for these two kinds of partition two corresponding algorithms are proposed. In the algorithm for parallel lines partition, our approach is to adaptively repartition the space by moving the vertical lines. The algorithm is introduced for determine how to move the given vertical line depends on the relative loads of sub-domains which have that line in common. And in the algorithm for quadrilateral grids partition, our approach is to repartition the space by moving the vertex in the grid. And the algorithm is focused on how to move a given vertex depends on the relative loads of sub-domains which have that vertex in common. These lines or vertex movement, as well as the transfer of any vehicle from one processor to another, can be carried out in parallel. Furthermore, we discuss the communication overheads induced by the algorithms, and the adaptability of each algorithm.

The rest of the paper is organized as follow: in section 2 we describe the algorithms for parallel lines partition and quadrilateral grids partition separately. In section 3 we present testing results of the two algorithms. Finally, section 4 provides the performance of the algorithms and some conclusion comments.

2 Algorithms

2.1 Load Balancing Algorithm Using Parallel Lines Partition

Using parallel vertical lines to partition the simulation region, just like using longitude to divide the map, has an obvious advantage in minimizing the message channel between processors. Using this type of partition, every processor has utmost two adjacent processors to communicate. And the decrease of message channels may cut down the communication time greatly at some occasions. Figure 1 illustrates the basic model of parallel lines partition.

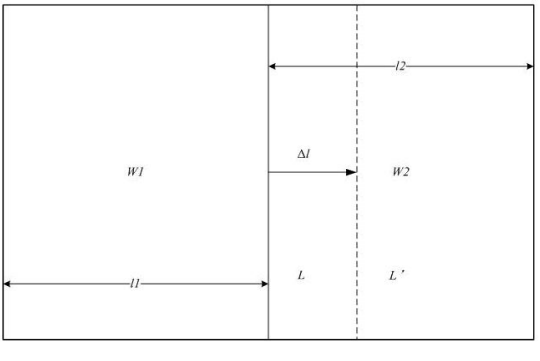


Fig. 1. This figure shows the basic model of parallel lines partition

We assume the total work load is *W*. And we use an arbitrary vertical line *L* to divide the work load into *W1* and *W2*, let $\overline{W}$ and $W_{\max}$ be the average work load and maximize work load among the processors separately. Then we define $I_l = 1 - \overline{W} / W_{\max}$ be the measure of imbalance of the workloads, and let I_{ideal} be the ideal imbalance. Obviously, if the workloads are evenly distributed, I_l would be quite small. Then we define another component:

$$\delta = \left| \frac{W_1 - W_2}{W} \right|$$

Here δ determines which side has the heavier load between two sub-domains. Then define:

$$\Delta l = (1 - \varepsilon)\delta \times l2 \qquad (0 < \varepsilon < 1)$$

Here Δl is the shift of *L* and ε is a parameter which controls how aggressively we seek to the balance.

We need to make several remarks:

1. When moving the line *L*, the work loads change monotonously and we can surely find an optimal position for balance.
2. When using more than one line, we can firstly adjust the lines which have an odd index, and then adjust the even lines.

The main steps of the algorithm are:

Step I. Calculate I_l according to the method mentioned above, if $I_l < I_{ideal}$, stop the algorithm, otherwise turn to S*tep II*.

Step II. For each odd line, calculate Δl.

Step III. Adjust *L* by Δl, and recalculate the work loads of the two sides.

Step IV. For each even line, calculate Δl.

Step V. Adjust *L* by Δl, and recalculate the work loads of each part, then turn to *Step I.*

2.2 Load Balancing Algorithm for Quadrilateral Grids Partition

Using quadrilateral grids or triangle grids to partition the simulation region is a widely accepted method. In this paper, we choose quadrilateral grids to divide the traffic map, and we describe the algorithm based on the basic partition model shown in figure 2.

As is illustrated in fig.2, Let *P* be an arbitrary grid point, and *E1*, *E4, E2, E3* are the neighbor grid points. Let the work loads of the four grid area which shares *P* be *W1*, *W2*, *W3*, *W4*, separately, and let $\overline{W}$, $W_{\max}$ be the average and maximize workload separately. Then the definition of imbalance is quit similar to that of prior algorithm: $I_l = 1 - \overline{W} / W_{\max}$. And let the ideal value of I_l be I_{ideal}.

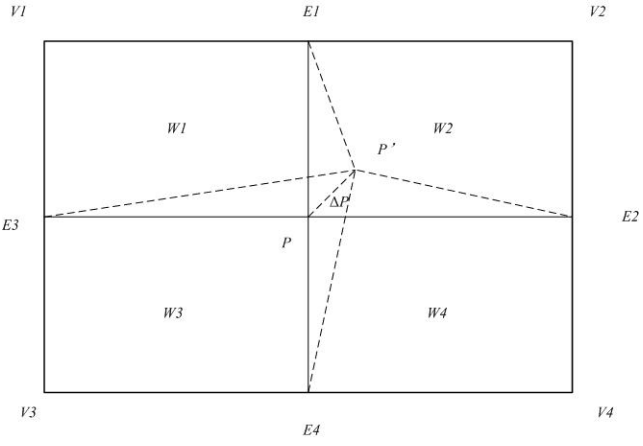


Fig. 2. The basic model of quadrilateral grids partition

Obviously, if we adjust P to P', $P'V1$, $P'V2$, $P'V3$, $P'V4$ will repartition the region, and so the work load of each sub-domain will be modified.

We define parameter $\delta 1$:

$$\delta 1 = \frac{(W1+W2)-(W3+W4)}{W1+W2+W3+W4}$$

Here $\delta 1$ presents which pair has the heavier load, the up pair or the down pair. Then we let *pE1* denotes the vector from *P* to *E1* and *pE4* be the vector from *P* to *E4*, then a vector $offset1$:

$$offset1 = \begin{cases} |\delta 1|\, pE1 & \delta 1 > 0 \\ |\delta 1|\, pE4 & \delta 1 < 0 \end{cases}$$

In the same way , using *W1+W3* and *W2+W4* , we define $\delta 2$, vector *pE2*, *pE3*, and $offset2$. Then the offset of point *P* is defined as follow:

$$\Delta P = (1-\varepsilon)\frac{offset1+offset2}{2}$$

Note that we move the point *P* by vector *pE1, pE2*, *pE3* and *pE4*, and the four sub-domains will remain convex.

Obviously, only by moving *P* will not necessarily reach the balance status. For example, if $W1 \approx W4 \gg W2 \approx W3$, we must adjust the locations of the points *E1*, *E2*, *E3*, *E4*. The movement of these points is quite similar to that of *P*. We take *E1* for example, define:

$$\delta_{E1} = \frac{W1-W3}{W1+W3}$$

And let *E1V1* be the vector from *E1* to *V1* and *E1V2* be the vector from *E1* to *V2*, and then the vector $offsetE1$ and the offset $\Delta E1$ are defined as follow:

$$offsetE1 = \begin{cases} |\delta 1|\, pE1 & \delta 1 > 0 \\ |\delta 1|\, pE4 & \delta 1 < 0 \end{cases}$$

$$\Delta E1 = (1-\varepsilon) offsetE1$$

The adjustment of *E2, E3* and *E4* is similar to that of *E1*, and we do not make any further discussion.

Based on the definition above, for a partition of $M \times N$ grid, we number all the points on the edges and inside of the region as $P_{ij}(0 \le i \le M, 0 \le j \le N)$ by its location. And all the work loads are numbered as $Wij(1 \le i \le M, 1 \le j \le N)$. For all the points, we apply a red-black coloring scheme to adjacent points to adjusting their location. We choose this type of scheme because in this strategy multi-processors can work synchronously without any conflict.

The algorithm is described as follow:

Step I. Calculate the work loads of all the sub-domains, and the imbalance degree I_l. If $I_l < I_{ideal}$, stop the algorithm, else turn to *Step II*.

Step II. For all the inside points, P_{ij} $(1 \le i \le M-1,\ 1 \le j \le N-1)$, choose the ones in odd rows and odd columns and the ones in even rows and even columns. Adjust these points themselves and the adjacent points.

Step III. For all the inside points, choose the ones in odd rows and even columns and the ones in even rows and odd columns. Adjust these points themselves and the adjacent points. And then turn to *Step I*.

3 Experimental Result and Analysis

We take the real traffic map of Shanghai China as the experimental task region. And we present the real distribution in a 1024×1024 bit map, in which each colored pixel stands for a number of vehicles. From white to red the color of each pixel stands for the density of vehicles from low to high. The balancing result based on the parallel lines partition is illustrated in figure 3.

Figure 4 illustrates the detail of the balancing procedure and the distribution of work loads between processors.

Figure 5 and figure 6 illustrate the balancing result based on a 4×4 quadrilateral grid partition.

From the experimental result we can see that using parallel lines to partition the domain, the balancing algorithm can gain a rather good result. But this type of partition may lead to big communication overheads between some processors. So in order to compare the algorithm, we design another model to estimate the communication overheads between processors.

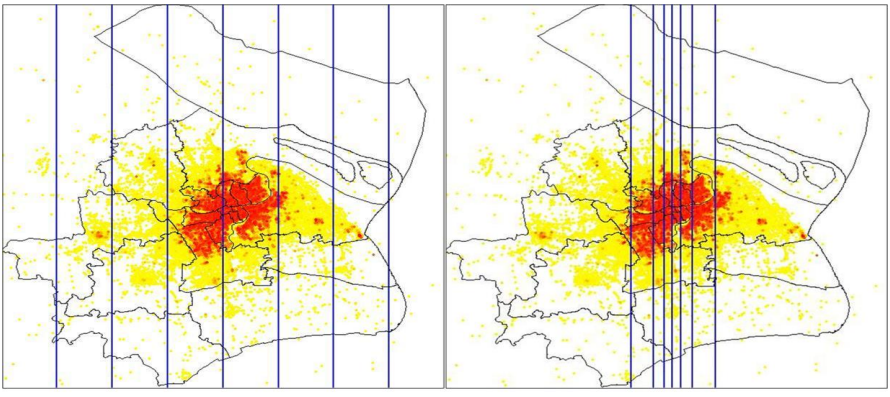


Fig. 3. This figure shows the status before balancing (a) and 100 algorithm cycles later (b) based on parallel lines partition

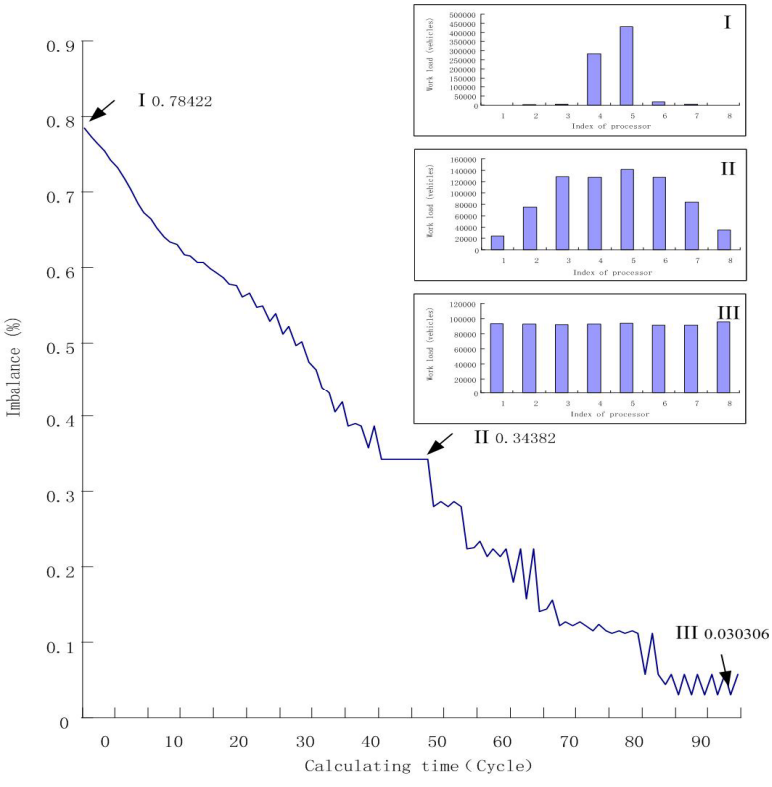


Fig. 4. This figure shows the relationship between the imbalance and calculating cycles ($\varepsilon = 0.9$) in 100 cycles based on parallel lines partition. Inset in this figure are three histograms showing the initial, midpoint, and final load distributions.

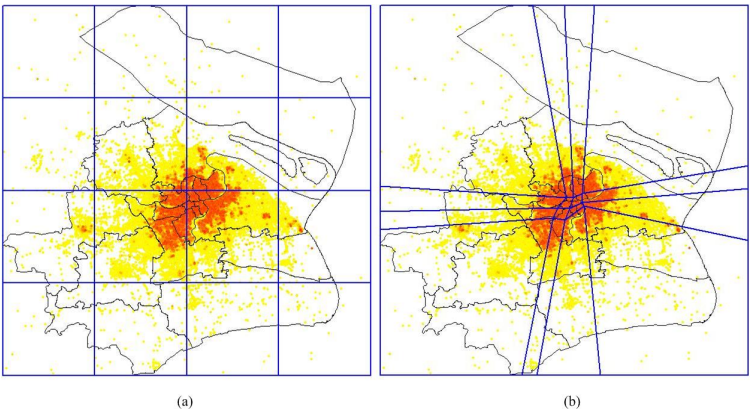


Fig. 5. This figure shows the status before balancing (a) and 32 algorithm cycles later (b) based on a 4×4 grid partition

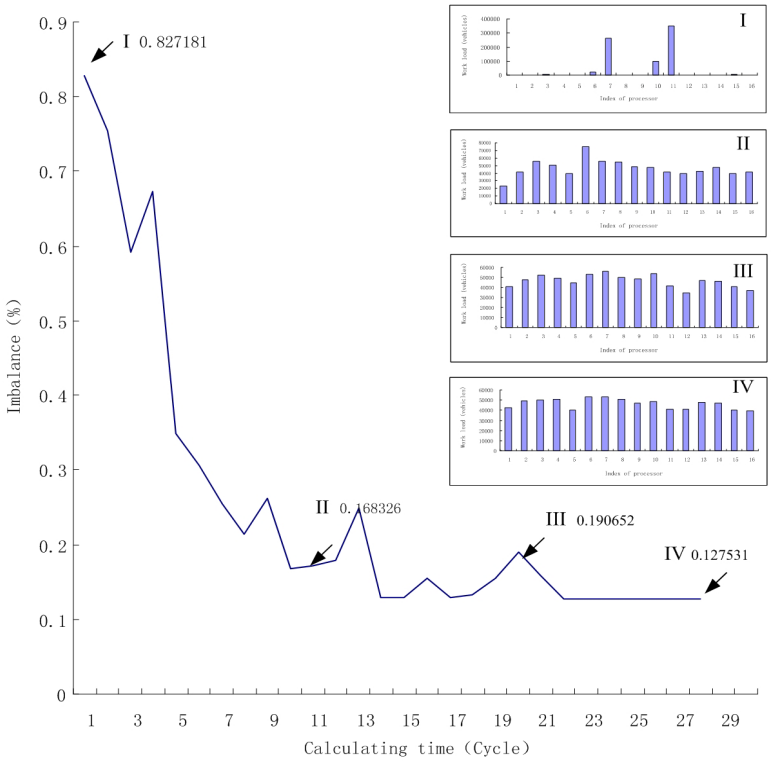


Fig. 6. This figure shows the relationship between the imbalance and calculating cycles ($\varepsilon = 0.9$) in 100 cycles based on a 4×4 quadrilateral grid partition. Inset in this figure are four histograms showing the initial, midpoints, and final load distributions.

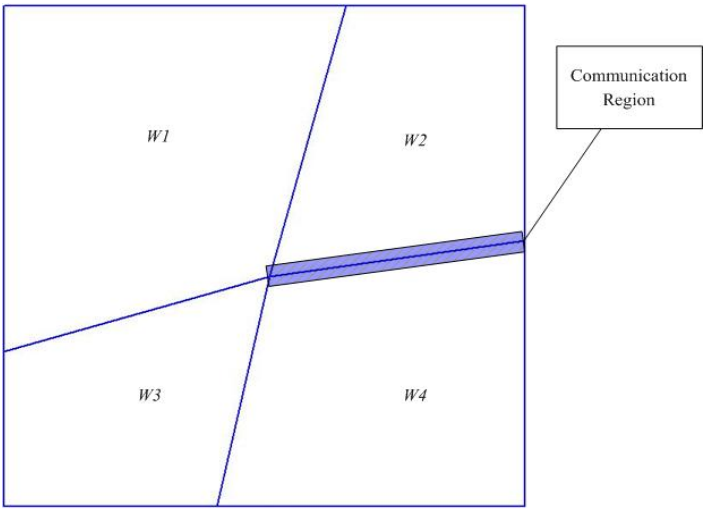


Fig. 7. This figure shows the basic communication overhead estimating model

In the parallel simulation of transportation, vehicles may run across the boundary lines and need to be transferred from one processor to another. So the communication overheads can be estimated by accounting the number of vehicles near the boundary lines. As is shown in figure 7, we name the adjacent area of a partition line a communication region, and by accounting the vehicles inside the area we estimate the approximate communication overhead for each processor. In the experiment we set the width of the communication region to two pixels in the bit map. Figure 8 illustrates the result on the communication overheads estimation.

Through calculating the amount of vehicles in the communication areas, we estimate the communication overhead between each pair of adjacent processors. And as

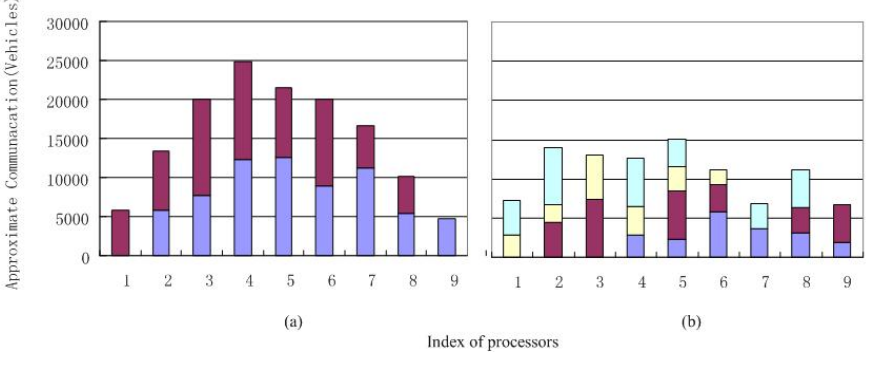


Fig. 8. This figure shows the communication overhead of 8 parallel lines partition (a) and 3×3 grid partition (b), different colors represent the communication overheads with different processors.

illustrated in figure 8, we use different colors to represent the communication with different processors. In figure 8(a), we can see the communication of each processor is composed of utmost two parts, but is much larger than that in figure8(b).

In figure 8(b), the communication overhead of processor 5 is composed of four parts, because it is in charge of the central sub-domain in the 3×3 grid. However the over communication head is not very large. This is because the four boundary lines of its sub-domain are quite short after balancing.

4 Conclusion

On the problem of dynamic load balancing in parallel transportation simulation, we propose an algorithm for parallel lines partition and an algorithm for quadrilateral grids partition. Based on the real traffic density distribution, we make several experiments to test the performance of each algorithm. The result shows that the algorithm using parallel lines partition can obtain a relative low imbalance in 100 algorithm cycles, and the algorithm using quadrilateral grids partition can obtain a relative high imbalance. But this does not necessarily mean that the former is more advisable and in the later experiment of comparing the communication overheads of an 8 parallel lines model and a 3×3 grid model. We find that the communication overhead of the former is much bigger than that of the later and is not adaptable in real application unless on a quit fast network.

References

1. O'Cearbhaill, E.A., O'Mahony, M.: Parallel implementation of a transportation network model. Journal. Parallel Distributed Computing 65, 1–14 (2005)
2. Nagel, K., Rickert, M.: Parallel implementation of the TRANSSIMS micro-simulation. Parallel Computing 27, 1611–1639 (2001)
3. Klefstad, R., Zhang, Y., Lai, M., Jayakrishnan, R., Lavanya, R.: A Distributed, Scalable, and Synchronized Framework for Large-Scale Microscopic Traffic Simulation. In: Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems, pp. 813–818 (2005)
4. Deng, Y., Peierlsy, R.F., Riveraz, C.: An Adaptive Load Balancing Method for Parallel Molecular Dynamics Simulations. Journal of Computational Physics 161, 250–263 (2000)
5. Rus, P., Tok, B., Mole, N.: Parallel computing with load balancing on heterogeneous distributed systems. Advances in Engineering Software 34, 185–201 (2003)
6. Genaud, S., Giersch, A., Vivien, F.: Load-balancing scatter operations for grid computing. Parallel Computing 30, 923–946 (2004)

Embedded System's Performance Analysis with RTC and QT

Fulong Chen[1,2] and Xiaoya Fan[1]

[1] School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China
chenfulong@mail.nwpu.edu.cn
[2] Department of Computer Science and Technology, Anhui Normal University, Wuhu 241000, China
fanxy@nwpu.edu.cn

Abstract. Maximal, minimal and mean response time are several performance metrics of embedded system. Some embedded systems have not real-time constraints, e.g. printer and router, but their mean case performance is more important, and some others have those constraints and need evaluating their utmost performance. In this paper, with real-time calculus and queuing theory, both mean and utmost case performances of embedded multitasking system are analyzed as referred target for design, verification, decision and optimization.

Keywords: embedded system, performance analysis, real-time calculus, queuing theory.

1 Introduction

With the development of computer theory and technology, more and more smart and embedded devices are applied in many areas, e.g. automobiles electronics, handheld game systems, industrial environments, telecommunication or fabrication equipments, aircraft electronics, medical systems, military applications, authentication systems, consumer electronics, smart buildings, robotics and so on. Embedded systems can be defined as information processing systems embedded into these enclosing products [1]. Hence, embedded computer has also been called the disappearing computer, which form the basis of the so-called pervasive computing era.

As shown Fig.1, Sensors receive signals or information from the physical environment, and send them to A/D converters or sample-and-holders, which convert continuous sequences of analog values to discrete sequences of digital values. The kernel of the whole embedded system-information processor processes those digital values and generates some digital results. These results can be sent to display device, which displays these results, and also be used to control the external environment through actuators, which convert digital signals to analog ones.

During developing of embedded system, it is very important to evaluate its performance. By evaluating memory requirements, timing properties, processor speed, bus utilization, etc., developers can find the bottleneck components so as to select appropriate hardware or software resources that meet the system's functional

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 569–579, 2007.

and non-functional requirements and cost low. Furthermore, by evaluating performance of actual embedded application system, testers can verify whether the performance meets the design requirements and constraints those are presented by customers.

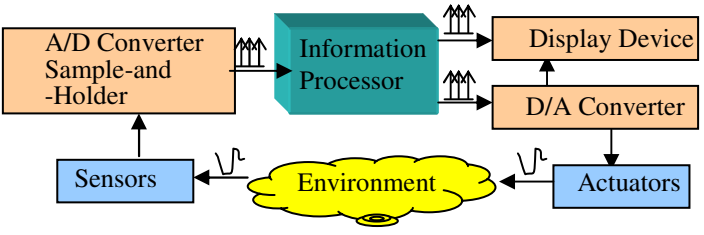


Fig. 1. Architecture of Embedded System

There are three solutions for evaluating performance of embedded system:

- *Measurement*-by executing some test benchmarks or instances, testers can get Execution Time (ET) and other performance parameters for evaluating the actual system. In the past years, many universities, institutes and corporations have been devoted to studying the technologies of benchmark performance for embedded system. Some performance evaluation tools and benchmarks, e.g. Dhrystone, EEMBC, NullStone, MediaBench, MiBench, etc., have been developed and released. With these tools and benchmarks, some performances of hardware and software of embedded system can be evaluated. However, many system functions and attributes are hard to be obtained by testing due to the variety of the architectures of embedded system.
- *Simulation*-by executing some programs for simulating the evaluated system and its load, monitoring members can get more precise performance parameters. But this solution's cost is higher when its model is constructed. At present, for embedded system, there are many simulation tools such as Carosse-Perf, SES Workbench [2], STRESS, schedalyzer, ARTS, PERTS [3], RTC Toolbox [8] and so on.
- *Analysis*-by making use of some mathematical theories and methods, analyzers can study and describe the relation of performance and system before design, and get loose performance metrics. According to this approach, its cost for modeling is the lowest. Real-Time Calculus (RTC) and Queuing Theory (QT) are two kinds of system level analysis tools for evaluating performance of real-time embedded system. Performance of embedded system includes best, worst and mean ones. RTC is fit for analyzing best performance and worst performance, and QT is more propitious to analyzing mean performance.

In section 2, Real-Time Calculus is introduced, and performance of embedded system in utmost state is analyzed. In section 3, basic queuing theory is introduced, and by making use of this basic theory and method, performance of embedded system in steady state is also analyzed.

2 Real-Time Calculus (RTC) and Performance Analysis of Embedded System in Utmost State

2.1 RTC

RTC is on basis of Linear System Theory, Calculus for Network, Adversarial Queuing Theory [7], and composed of input/output event streams model, resource (service) model and processing model (Fig. 2)

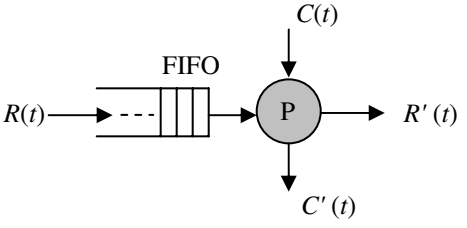


Fig. 2. Modeling of Real-Time Calculus System

- $R(t)$:number of events that arrived in time interval [0, t),represents input event stream function;
- $R'(t)$:number of events that left in time interval [0, t), represents output event stream function;
- $C(t)$:maximal number of events that could have been processed by processor P in time interval [0,t], represents available resources or services function;
- $C'(t)$: maximal number of events that processor P could still process by in time interval [0,t], represents remaining resources or services function.

If the description of resource/service and event streams is available, then $C(t)$ and $R(t)$ are determinate. So

$$R'(t) = \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\} \tag{1}$$

$$C'(t) = C(t) - R'(t) \tag{2}$$

2.2 Performance Analysis of Embedded System in Utmost State

Provided that, in the interval $[0, \Delta]$, $\alpha^u(\Delta)$ represents maximum number of arriving events, $\alpha^l(\Delta)$ represents minimum number of arriving events, $\beta^u(\Delta)$ represents maximum number of available services (resources), and $\beta^l(\Delta)$ represents minimum number of available services (resources), according to RTC, the following processing model can be got:

$$\alpha'^u(\Delta) = f_\alpha{}^u(\alpha^u, \alpha^l, \beta^u, \beta^l) \tag{3}$$

$$\alpha'^{l}(\Delta) = f_{\alpha}{}^{l}(\alpha^{u}, \alpha^{l}, \beta^{u}, \beta^{l}) \tag{4}$$

$$\beta'^{l}(\Delta) = f_{\beta}{}^{l}(\alpha^{u}, \alpha^{l}, \beta^{u}, \beta^{l}) \tag{5}$$

$$\beta'^{u}(\Delta) = f_{\beta}{}^{u}(\alpha^{u}, \alpha^{l}, \beta^{u}, \beta^{l}) \tag{6}$$

Here, $f_{\alpha}{}^{u}$, $f_{\alpha}{}^{l}$, $f_{\beta}{}^{u}$ and $f_{\beta}{}^{l}$ depend on real processing model. For example, in greedy shaper component model, $\beta^{u}(\Delta) = \beta^{l}(\Delta) = \sigma$,so $\alpha'^{u}(\Delta) = \inf_{0 \le \lambda \le \Delta} (\alpha^{u}(\Delta - \lambda) + \sigma)$ [9,10]. Through this method, with RTC Toolbox [8], for a known processing model, its arrival curves [α^{u}, α^{l}], leave curves [α'^{u}, α'^{l}], service curves [β^{u}, β^{l}] and availability curves [β'^{u}, β'^{l}] are obtained. If the performance doesn't meet the requirements of actual embedded application system, designers need adjust input events stream so as to change arrival curves [α^{u}, α^{l}] and reconfigure resources so as to change service curves [β^{u}, β^{l}], or, reconstruct new processing model so as to change $f_{\alpha}{}^{u}$, $f_{\alpha}{}^{l}$, $f_{\beta}{}^{u}$ and $f_{\beta}{}^{l}$.

3 Queuing Theory and Performance Analysis of Embedded System in Steady State

3.1 Basic Queuing Theory

In Queuing theory, a random service system is taken into account. By studying the probability parameters of customers waiting in queue, analyzers can get the performance of queuing system in steady state. A queuing system can be described as Fig.2. Any customer that wants to pass through the system need finish the following processes: arriving, waiting, served and leaving. The interval between two customers arriving orderly satisfy Determinate Length Distribution (DLD), Negative Exponential Distribution (NED, called Poisson stream), Erlang Distribution (ED) or general random distribution. Customers wait in the queue according to wait or loss rule. In wait rule, customers are served according to First Come First Serve (FCFS), Last Come First Serve (LCFS), Priority Serve (PS), general random serve, etc. There is one server or multi-servers in the system. The former is called Singer Server Queuing System (SSQS), and the latter goes by the name of Multiple Servers Queuing System (MSQS). The served time length of each customer may be subject to DLD (*D*), NED (*M*), k ranks ED (E_k) or general random distribution (*G*). In 1950s,D.G.Kendall introduced *A/B/C/n* queue model- *A* stands for input stream, *B* stands for served time, *C* stands for number of servers, n stands for queue length. For instance, for *M/M/1/*∞ queue, the first M means that customers arrive in Poisson stream, the second *M* stands for Markovian or memoryless (exponential) served time

distribution, 1 stands for only one server, and ∞ is indicated that the length corresponding to the queue is limitless. In Queuing system, mean queue length L, mean steady queue length L_q, mean stay (or response) time T and mean wait time T_q are four performance metrics in common use for performance analysis. Queuing system has been used for multi-process controlling [11], VoIP QoS analysis [12], Evaluation for UML [13], multiple access method [14], simulator [15], dispatching strategy [16], railway computer interlocking system performance evaluation [17], estimation of the storage space [18], Evaluation of anonymity providing techniques [19], staff software maintenance centers [20], etc.

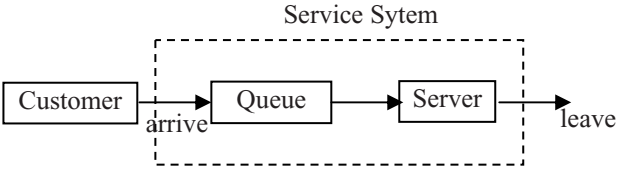


Fig. 3. Specification for Service System

3.2 Embedded Multitasking Sequential Processing Model

An embedded system and some tasks can be considered as a Markovian queuing process, or a birth and death process. The embedded system is the server, and tasks, which are waiting in a queue to get services, are customers. Each task, located in the memory or external environment and requesting service, is considered as a customer arrival in queue. The rate of arrival of the tasks is the birth rate and the rate of their leaving the system after finishing their services is the death rate. It is assumed that the death rate can be influenced by some queuing control strategies but the birth rate cannot. It is the goal to change the death rate, or the users' service rate, in a way that the number of served tasks would be maximized after some finite time.

Assumed that tasks arrival rate is subject to Poisson distribution, the processor processes tasks according to FCFS, the spent time is subject to NED, and then an *M/M/1/*∞ queue model can be constructed (Fig.4).

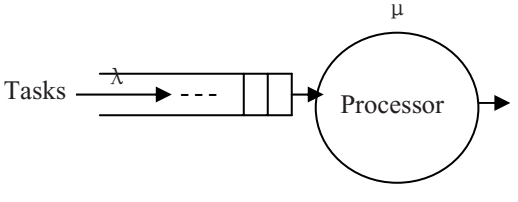


Fig. 4. Embedded Multitasking Sequential Processing Model

In this model, the following parameters are defined:

- λ—arrival rate stands for number of tasks arriving in unit time;
- μ—service rate or departure rate stand for number of tasks processed in unit time;

- ρ—server utilization or service intensity , ρ=λ/μ.

λandμof embedded system can be solved, see [21]. Then, mean response time T of tasks can be got through Little formula [22]:

$$T = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu - \lambda} \tag{7}$$

3.3 Embedded Multitasking Non-preemptive Model

In Fig.4, if there are n ranks of different priorities for tasks, each rank of tasks are Poisson stream, arrival rate is λ_i, $1 \le i \le n$, processing time is not determinate, but mean processing time $1/\mu_i = E[s_i]$ and $\gamma_i = E[s_i^2]$ can be acquired, kindred tasks are processed according to FCFS, different kinds of tasks are processed according to non-preemptive rule, then that model becomes a Non-Preemptive *M/G/1/*∞ Model.

Given $\lambda = \sum_{i=1}^{n} \lambda_i$, $1/\mu = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda \mu_i}$, $\gamma = \sum_{i=1}^{n} \frac{\lambda_i \gamma_i}{\lambda}$, $u_j = \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i}$, then mean wait time of j-th rank of tasks:

$$T_q(j) = \frac{\lambda \gamma}{2(1-u_{j-1})(1-u_j)} \tag{8}$$

And mean response time:

$$T(j) = T_q(j) + \frac{1}{\mu_j} \tag{9}$$

Mean wait time of whole system:

$$T_q = \sum_{j=1}^{n} \frac{\lambda_j T_q(j)}{\lambda} \tag{10}$$

And mean response time of whole system:

$$T = \sum_{j=1}^{n} \frac{\lambda_j T(j)}{\lambda} \tag{11}$$

3.4 Embedded Multitasking Preemptive and Non-blocked Model

In Preemptive and Non-blocked Model, high rank of tasks can snatch the processor from low rank of tasks, which will go back to the queue and wait for next process. Then the system becomes a feedback Jackson Queuing Network [22](Fig.4 (a)). Provided γ is arrival rate of whole system, q is the probability of preemption, μ is processor's service rate, then tasks arrival rate λ of processor queue can be obtained according to Equation (12):

$$\lambda = \gamma + q\lambda \Rightarrow \lambda = \frac{\gamma}{1-q} \tag{12}$$

The processor utilization is

$$\rho = \frac{\lambda}{\mu} = \frac{\gamma}{\mu(1-q)} \tag{13}$$

So mean number of tasks of whole system is:

$$L = \frac{\rho}{1-\rho} = \frac{\gamma}{\mu(1-q)-\gamma} \tag{14}$$

According to Little formula, mean respone time of whole system is:

$$T = \frac{L}{\gamma} = \frac{1}{\mu(1-q)-\gamma} \tag{15}$$

It is obvious that relation between T and q is showed by Fig. 4(b), and while q=0,this model becomes that one of Fig.3; while $q \rightarrow 1-\gamma/\mu$,$T \rightarrow \infty$.

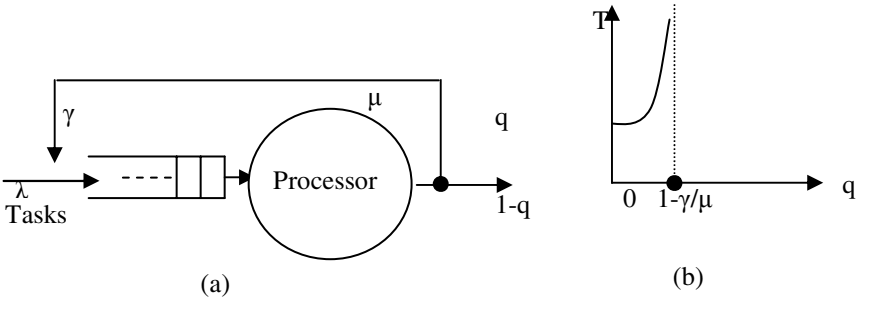


Fig. 5. Embedded Multitasking Preemptive and Non-blocked Model Model

3.5 Embedded Multitasking Non-preemptive and Blocked Model

Now see non-preemptive and blocked model (Fig.5). Tasks arrive at the system in arrival rate γ. Processor processes them in service μ_1.Tasks can be blocked by so-called block task during processing as a result of some I/O operations or synchronization. Supposed that blocked probability is q, and rate of release from block is μ_2, blocked tasks will go back to processor queue and wait for next process after completing their I/O operations or synchronization, then real arrival rateλ_1 of processor queue and real arrival rateλ_2 of block queue have the following relation:

$$\lambda_1 = \gamma + \lambda_2 \tag{16}$$

$$\lambda_2 = q\lambda_1 \tag{17}$$

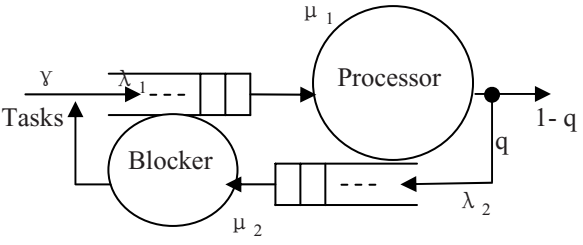


Fig. 6. Embedded Multitasking Non-preemptive and Blocked Model

So,

$$\lambda_1 = \frac{\gamma}{1-q} \tag{18}$$

$$\lambda_2 = \frac{q\gamma}{1-q} \tag{19}$$

Accordingly, processor service intensity ρ_1 and block service intensity ρ_2 are:

$$\rho_1 = \frac{\lambda_1}{\mu_1} = \frac{\gamma}{\mu_1(1-q)} \tag{20}$$

$$\rho_2 = \frac{\lambda_2}{\mu_2} = \frac{q\gamma}{\mu_2(1-q)} \tag{21}$$

Mean length of processor queue L_1 is:

$$L_1 = \frac{\rho_1}{1-\rho_1} = \frac{\gamma}{\mu_1(1-q)-q\gamma} \tag{22}$$

Mean length of blocker queue L_2 is:

$$L_2 = \frac{\rho_2}{1-\rho_2} = \frac{q\gamma}{\mu_2(1-q)-\gamma} \tag{23}$$

Mean length of whole system queue is:

$$L = L_1 + L_2 = \frac{\gamma}{\mu_1(1-q)-q\gamma} + \frac{q\gamma}{\mu_2(1-q)-\gamma} \tag{24}$$

So, mean response time of whole system is:

$$T = \frac{L}{\gamma} = \frac{1}{\mu_1(1-q) - q\gamma} + \frac{q}{\mu_2(1-q) - \gamma} \tag{25}$$

Range of q is:

$$0 \le q < \min\left(\frac{\mu_1}{\mu_1 + \gamma}, 1 - \frac{\gamma}{\mu_2}\right) \tag{26}$$

3.6 Embedded Multitasking Preemptive and Blocked Model

Both preemption and block are considered, see Fig.6.Tasks arrive at the system in γ. Processor processes them in μ_1.Low rank of tasks can be preempted by high ones, and also be blocked due to some I/O operations or synchronization. On the assumption that preempted probability is q_1, blocked probability is q_2, and rate of release from block is μ_2.After those interruptions, tasks will go back to processor queue and wait for next process. Then, real arrival rateλ_1 of processor queue and real arrival rate of block queueλ_2 have the following relation:

$$\lambda_1 = \gamma + \lambda_1 q_1 + \lambda_2 \tag{27}$$

$$\lambda_2 = \lambda_1 q_2 \tag{28}$$

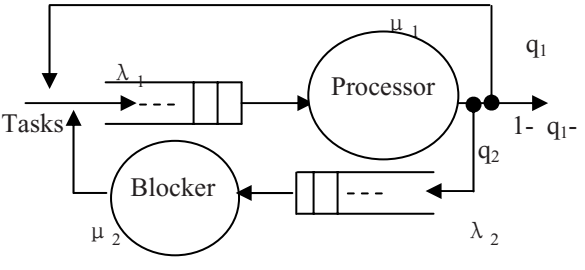


Fig. 7. Embedded Multitasking Preemptive and Blocked Model

So,

$$\lambda_1 = \frac{\gamma}{1 - q_1 - q_2} \tag{29}$$

$$\lambda_2 = \frac{q_2\gamma}{1 - q_1 - q_2} \tag{30}$$

Mean response time of whole system is:

$$T = \frac{L}{\gamma} = \frac{L_1 + L_2}{\gamma} = \frac{\frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2}}{\gamma} = \frac{\frac{\frac{\lambda_1}{\mu_1}}{1-\frac{\lambda_1}{\mu_1}} + \frac{\frac{\lambda_2}{\mu_2}}{1-\frac{\lambda_2}{\mu_2}}}{\gamma} \tag{31}$$

$$= \frac{1}{\mu_1(1-q_1-q_2)-\gamma} + \frac{q_2}{\mu_2(1-q_1-q_2)-q_2\gamma}$$

To all appearances, non-preemptive and blocked model ($q_1 \to 0$), preemptive and non-blocked model ($q_2 \to 0$) and sequential processing model ($q_1 \to 0$, $q_2 \to 0$) are three special cases of preemptive and blocked model.

4 Conclusions and Future Work

With RTC and QT, utmost and mean performance of embedded system can be assured loosely and quickly so as to select appropriate design scheme, verify and optimize system. RTC tool has been offered [8].However QT tool for embedded system is not available.In the future, QT tool of different models will be submitted for evaluating performance of embedded system.

Acknowledgments. This paper is supported by the National Technology Research and Development 863 Program of China (No. 2005AA1Z1193),the National Natural Science Foundation of China (No. 60573143) ,the Soft Science of Anhui Province of China(No.06035021) and Youth Fund of Anhui Normal University(No. 2006xqn54).

References

1. Marwedel, P.: Embedded System Design, pp. 1–7. Springer, Heidelberg (2006)
2. Castelpietra, P., Song, Y.Q., Lion, F.S., et al.: Analysis and Simulation Methods for Performance Evaluation of a Multiple Networked Embedded Architecture. IEEE Transactions on Industrial Electronics 49(6), 1251–1264 (2002)
3. Audsley, N.C., Burns, A., Richardson, M.F., et al.: STRESS:a Simulator for Hard Real-time Systems. Software-Practice and Experience 24(6), 543–564 (1994)
4. Li, Y.T.S., Malik, S.: Performance Analysis of Embedded Software Using Implicit Path Enumeration. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems 16(12), 1477–1487 (1997)
5. Malik, S., Martonosi, M., Li, Y.T.S.: Static Timing Analysis Of Embedded Software. In: 34th Design Automation Conference, pp. 147–152 (1997)
6. Lee, J.Y., Park, I.C.: Timed Compiled-Code Functional Simulation of Embedded Software for Performance Analysis of SOC Design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22(1), 1–149 (2003)

7. Thiele, L., Chakraborty, S., Naedele, M.: Real-time calculus for scheduling hard real-time systems. In: 2000 IEEE International Symposium on Circuits and Systems, pp. 101–104. IEEE Computer Society Press, Los Alamitos (2000)
8. ETH Zurich.Real-Time Calculus(RTC) Toolbox, http://www.mpa.ethz.ch/rtctoolbox,2007-4-1
9. Thiele, L.: Performance Analysis of Distributed Embedded System. Technical report,ARTIST 27 UNU-IIST Spring School in Xi'an,China (April 3-25, 2006)
10. Thiele, L.: Introduction to Real-Time Calculus. Technical report,ARTIST 27 UNU-IIST Spring School in Xi'an,China (April 3-25, 2006)
11. Egerstedt, M., Wardi, Y.: Multi-Process Control Using Queuing Theory. 41st IEEE Conference on Decision and Control 2, 1991–1996 (2002)
12. Pitts, J.M., Wang, X., Yang, Q., Schormans, J.A.: Excess-rate queuing theory for M/M/1/RED with application to VoIP QoS. Electronics Letter 42(20), 1188–1189 (2006)
13. Perdos, A., Chatzigeorgiou, A., Stephanides, G.: Evaluation of a Queuing Theory and Systems Modeling Course Based on UML. In: 6th International Conference on Advanced Learning Technologie, pp. 507–509 (2006)
14. Jamshidifar, A.A., Khorram, E., Afshar, A.: Applying queuing theory to multiple access method for store and forward satellites. In: 2nd International Conference on Recent Advances in Space Technologies, pp. 416–419 (2005)
15. Granados, D., Garcia, A.J.: A Web based simulator on elementary queuing theory. In: 5th International Conference on Information Technology Based Higher Education and Training, pp. 83–86 (2004)
16. Zong, Q., Xing, G., Chen, D., Ya, S.: The queuing theory based research of dispatching strategy for elevator group control system during up-peak. 5th World Congress on Intelligent Control and Automation 6, 5307–5311 (2004)
17. Guo, J., Zhu, C., Yang, Y.: Performance evaluation of railway computer interlocking system based on queuing theory. In: 4th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 420–423 (2003)
18. Yang, C.-W., Huang, K.-S., Yu, G., Jan, D.-Y.: Using queuing theory to estimate the storage space of stocker in automated material handling systems. In: Semiconductor Manufacturing Technology Workshop, pp. 102–104 (2002)
19. Kesdogan, D.: Evaluation of anonymity providing techniques using queuing theory. In: 26th Annual IEEE Conference on Local Computer Networks, pp. 316–322. IEEE Computer Society Press, Los Alamitos (2001)
20. Antoniol, G., Casazza, G., Di Lucca, G.A., Di Penta, M., Rago, F.: A queue theory-based approach to staff software maintenance centers. In: 2001 IEEE International Conference on Software Maintenance, pp. 510–519 (2001)
21. Luo, G., Zhao, Z., Zhao, H.: Testing and Evaluation of Network Capability with Queueing Theory in a Embedded System. Journal of Shenyang Normal University (Natural Science), China 23(1), 54–56 (2005)
22. Lin, C.: Computer Nerwork and Computer System Perforance Evaluation, pp. 224–260. TsingHua University Press, China (2001)

Scheduling Meetings in Distance Learning*

Jian Wang, Changyong Niu, and Ruimin Shen

Department of Computer Science and Engineering
Shanghai Jiaotong University, Shanghai 200030, China
{jwang,cyniu,rmshen}@sjtu.edu.cn

Abstract. Peer-to-peer technique becomes mature gradually, and multiple domain-involved applications emerge, such as IPTV, distance learning, chatting network. In the context of distance learning, small scale of teachers would serve large scale of geographically located students. Normally, knowledge points are associated with different difficulty levels. And students usually are interested in varied subsets of knowledge points. Also teachers are capable of serving knowledge point subsets. The objective to schedule meeting among students and teachers according to their respective interests and capabilities is to reduce total learning duration. After formulating meeting schedule as Integer Programming problem, this paper proposes three heuristic algorithms to approximate the optimal solution. To the best of our knowledge, such problem is firstly investigated in distance learning context. Performance evaluation demonstrates their behaviors and *PKPA* algorithm excels two others substantially.

Keywords: Scheduling, Timetable, Integer Programming, Heuristics, Distance Learning.

1 Introduction

Scheduling is of common combinatorial optimization and planning problem, and finds usage in almost every resource-constrained scenarios. The timetable scheduling, one of important scheduling problems, exists in realistic life, such as lecture, transportation, examination, meeting. Due to heavy problem complexity and involved domain context, such problem is widely solved with heuristics.

Peer-to-Peer technique has been thoroughly studied in the past decade. And multiple domain-involved applications emerge such as IPTV, distance learning, chatting network, content-based service. The Peer-to-Peer overlay distributes learning content, consisting of audio, video, and handwriting, to large population of consumers efficiently. Consequently, distance learning achieves big progress in terms of scale of service capability. Among involved students and teachers in distance learning, there exists scheduling issue for efficient resource usage.

Knowledge points in distance learning are normally associated with difficulty levels. Students request different subset of them due to interest or difficulty

* This work is supported by the NFSC under Grant No.60672066, China.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 580–589, 2007.

level. In addition, the teachers do also specialize in subset of knowledge points whose size is often large. The objective of scheduling meetings among involved students and teachers according to their respective interests and capabilities is to reduce total learning duration. Of course, such scheduling strategy can apply into virtual lab or virtual collaboration on the Internet, too.

To the best of our knowledge, scheduling meetings in context of distance learning is firstly investigated. The remainder of this paper is organized as follows. Section 2 reviews representative works on timetable scheduling. Section 3 formally formulates scheduling meeting problem and three heuristic algorithms are proposed consequently in section 4. The performance of algorithms are numerically evaluated in section 5. The paper concludes in section 6 with future work.

2 Related Work

As scheduling is concerned, the most relevant works are timetable ones. Tabu [1] deals with problem of assigning teachers to courses in a secondary school. Such timetable is to be built when teaching assignments are not fixed. It considers the characteristics of the school week, finite teachers and rooms, individual subject requirement, prerequisites, as well as characteristics and regulations of country-specific education system into account and finds a schedule for a set of meetings between teachers and groups of students over a set of time periods using tabu searching algorithm. Timetable is so difficult to solve due to large search space and highly constrained requirements. Thus, works [2] and [3] strive to approximate the optimal solution for timetable problem by genetic algorithms.

TGA [4] presents two-phase genetic algorithm to solve timetable problem for universities, and it uses two populations for class scheduling and room allocation, respectively. Consequently, it achieves better performance than simple genetic algorithm. Work [5] utilizes Particle Swarm Optimization technique to solve the discrete problem of timetable scheduling. And PSO performs well in discrete problem. Furthermore, the timetable problem is solved with efficient heuristic numerical algorithm in [6]. The main objective of timetable is to find feasible time slots with respect to multiple constraints.

TCDMP [7] is a Timetable-Constrained Distance Minimization Problem, which is a sports scheduling problem applied for tournaments and the total travel distance on individual teams must be minimized. MICSP [8] tends to address curriculum planning problem, which is defined as constructing a set of courses for each semester - over a sequence of semesters - in order to satisfy the academic requirements such as for undergraduate university degree. Obviously, both have different objectives to be optimized compared to timetable works. Interestingly, scheduling meetings in distance learning also differs from existing works due to intrinsic objective. As students interested in same knowledge point can be served by one teacher, where such sharing is easy in peer-to-peer overlay, the objective in this paper is to minimize total learning duration with relatively few constraints. The total learning duration denotes the time period from the instant the learning begins to the time all students complete learning.

3 Problem Formulation

In distance learning, activities are often conducted in *group* form. The group is formed by students with common interest, as well as supervised teachers. Given a large set of knowledge points, students involved with common knowledge point form a group. In collaborated virtual experiment scenario, specific experiment is correlated with student subsets as well as teachers. Thus, group is common form in distance education, especially when number of overall students is large. Due to finite number of teachers, it is imperative to schedule learning groups in a sequence time-efficiently.

3.1 Assumptions and Constraints

For formulating scheduling meeting problem easily, the notations and assumptions are introduced as following.

- $\diamond$ P: set of knowledge points, and $P = \{p_1, p_2, \cdots, p_l\}$
- $\diamond$ T: set of teachers, and $T = \{t_1, t_2, \cdots, t_m\}$
- $\diamond$ S: set of students, and $S = \{s_1, s_2, \cdots, s_n\}$
- $\diamond$ X_{t_i}: subset of knowledge points that can be served by teacher t_i, and $X_{t_i} \subseteq P, 1 \leq i \leq m$
- $\diamond$ Y_{s_j}: subset of knowledge points that is requested by student s_j, and $Y_{s_j} \subseteq P, 1 \leq j \leq n$
- $\diamond$ Each knowledge point $p_k, 1 \leq k \leq l$ incurs same unit learning time

Assuming student in group interacts with supervised teacher during learning session, it is useless to prerecord lectures of knowledge point set such that student requests content of his favorites on demand, consequently. Thus, the constraints of scheduling meetings are given.

- $\diamond$ Each teacher t_i can serve at most one knowledge point at any time.
- $\diamond$ Each student s_j can enjoy at most one knowledge point at any time.

The problem to investigate is: Given knowledge point set P, teacher set T and their corresponding serving capabilities $\{X_{t_i}, 1 \leq i \leq m\}$, student set S and their corresponding knowledge requests $\{Y_{s_j}, 1 \leq j \leq n\}$, how does the scheduling strategy arrange students and teachers to study knowledge point together (i.e. meeting) in minimum total learning time.

3.2 Formulation

A decision variable z^r_{ijk} is introduced to denote whether student s_j can enjoy knowledge point p_k served by teacher t_i at time round r

$$z^r_{ijk} = \begin{cases} 1 & \text{student } s_j \text{ enjoys } p_k \text{ served by teacher } t_i \text{ at time round } r \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Consequently, the main objective is to minimize number of rounds for scheduling all involved teachers and students to complete all learning requests.

$$\text{minimize } R \tag{2}$$

Subject to

$$\sum_{j=1}^{n}\sum_{k=1}^{l} z_{ijk}^{r} \leq 1, \forall i \in \{1,2,\cdots,m\}, \forall r \in \{1,2,\cdots,R\} \tag{3}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{l} z_{ijk}^{r} \leq 1, \forall j \in \{1,2,\cdots,n\}, \forall r \in \{1,2,\cdots,R\} \tag{4}$$

$$\bigcup_{i=1}^{m}\bigcup_{k=1}^{l}\bigcup_{r=1}^{R}\{p_k | z_{ijk}^{r} = 1\} = Y_{s_j}, \forall j \in \{1,2,\cdots,n\} \tag{5}$$

$$\bigcup_{j=1}^{n}\bigcup_{k=1}^{l}\bigcup_{r=1}^{R}\{p_k | z_{ijk}^{r} = 1\} \subseteq X_{t_i}, \forall i \in \{1,2,\cdots,m\} \tag{6}$$

$$|\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\bigcup_{r=1}^{R}\{r | z_{ijk}^{r} = 1\}| \leq \tau, \forall k \in \{1,2,\cdots,l\} \tag{7}$$

$$\begin{aligned} z_{ijk}^{r} \in \{0,1\}, \forall r \in \{1,2,\cdots,R\}, \forall i \in \{1,2,\cdots,m\}, \\ \forall j \in \{1,2,\cdots,n\}, \forall k \in \{1,2,\cdots,l\} \end{aligned} \tag{8}$$

The formulation is a comprehensive integer linear programming. Equation (3) reflects the constraint that each teacher can serve at most one knowledge point at any time, and that of student in Equation (4). Equation (5) declares that individual learning request set Y_{s_j} must be satisfied. In addition, Equation (6) shows that teacher serves knowledge set no more than his capability X_{t_i}. Equation (7) enforces cost efficiency as the one knowledge point can at most be taught τ times. At last, Equation (8) indicates that the formulation is an integer programming. Since such problem is intrinsically complex, this paper resorts to heuristic algorithms to approximate the optimal solution.

4 Heuristic Algorithms

A simple example is given for above formulation before proposing three heuristic algorithms. In Figure 1, there are four students $\{s_1, s_2, s_3, s_4\}$, three knowledge points $\{p_1, p_2, p_3\}$, as well as three teachers $\{t_1, t_2, t_3\}$. Student requests and teacher capabilities are reflected by those edges in graph. For instance, student s_3 requests knowledge point set $\{p_1, p_2\}$, while teacher t_2 could serve knowledge point set $\{p_2, p_3\}$.

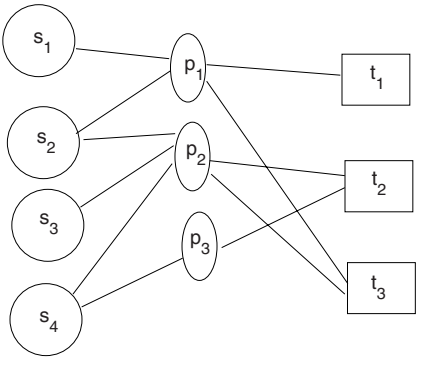


Fig. 1. An example relation graph for knowledge point learning

As scheduling meeting is concerned, there requires 2 rounds to complete knowledge learning process. In round 1, teacher t_2 serves the knowledge point p_3 that is interested by students s_4, while teacher t_1 serves p_1 that is enjoyed by s_1, s_3. Then in round 2, teacher t_2 serves p_2 that is interested by s_2, s_3, s_4. Obviously, there may exist multiple scheduling arrangements corresponding to same number of rounds. Such phenomenon contributes finding scheduling arrangement of minimum time quickly.

As size of students and knowledge points becomes larger, it is impracticable to find the optimal solution for the problem due to large computation load. Thus, three heuristic algorithms are proposed to approximate optimality quickly. They are feeded with same input and derive round number R individually. For clarity, the input is described as: knowledge point set P, student set S and corresponding request $\{Y_{s_j}\}$, teacher set T and corresponding capabilities $\{X_{t_i}\}$. For convenience, the edge between student s_j and his knowledge request $\{Y_{s_j}\}$ is uniquely labeled by each element in $\{Y_{s_j}\}$. Similarly, such labeling strategy applies to edges between teacher t_i and capability $\{X_{t_i}\}$.

In brief, three following algorithms are greedy-based. The τ-constraint defined in Equation (7) is relaxed in algorithms. Each iteration attempts to maximize number of students that can be served, although PSA, PTA, PKPA start iteration at position of student, teacher and knowledge point, respectively. In addition, each algorithm contains two layers of loop. The outer loop guarantees each individual student to be satisfied. And inner loop attempts to maximize served students in one round.

4.1 Preferential-Student Algorithm (PSA)

The idea is that in each iteration, first find student s_j with maximum knowledge points pending for studying. Then among those points, choose the one p_k that can be served by maximum available teachers. Finally, select the teacher t_i with minimum capability among corresponding teachers.

1: $E = \cup_{j=1}^{n} Y_{s_j}, round = 0$
2: WHILE $E \neq \emptyset$
3: $D = \{s | s \in S, Y_s = \emptyset\}, S = S - D$
4: $A = \emptyset, B = \emptyset, C = \emptyset, W = \emptyset$
5: WHILE $S - A - B \neq \emptyset$
6: $s = argmax_{x \in S-A-B} |Y_x|$
7: $p = argmax_{x \in Y_s - W} |\Omega(x)|$
8: IF $\Omega(p) == \emptyset$
9: $B = B \cup \{s\}$
10: CONTINUE
11: END
12: $t = argmin_{z \in \Omega(p)} |X_z|$
13: $C = C \cup \{t\}, V = \{s | s \in S - A - B, p \in X_s\}$
14: $A = A \cup V, W = W \cup \{p\}$
15: $Y_z = Y_z - \{p\}, \forall z \in V$
16: END
17: $E = \cup_{j=1}^{n} Y_{s_j}$
18: $round = round + 1$
19: END
20: RETURN *round*

Fig. 2. Preferential-Student Algorithm

Definition 1. *Let* $\Omega(p) = \{t | p \in X_t, t \in T, t$ *is available*$\}$ *denote the teacher subset, where each is available and is capable to serve knowledge point* p*.*

In Figure 2, E represents collection of pending student requests and S denotes students that still have requests. The S will be divided into two set A and B. A contains the students that can learn in current round, while B contains those not. Step 6 selects the student s with largest size of request set. Step 7 chooses the knowledge point p of s that has maximum available teachers. If such p does not exist, the student s joins into set B. Otherwise, the teacher of minimum capability t corresponding to p is chosen. Consequently, the unavailable teacher set C, knowledge point served W, the learning student set A as well as knowledge request of necessary students Y_z are updated in sequence. Whenever $S - A - B == \emptyset$ (i.e. the current students are divided into groups), the inner loop stops. Thus, E is updated and round is increased by one before next loop.

4.2 Preferential-Teacher Algorithm (PTA)

This algorithm takes inverse design direction compared to PSA. The idea is that in each iteration, first find maximum capability teacher t_i, then select knowledge point $p_k \in X_{t_i}$ with maximum interested students.

In Figure 3, E represents the pending knowledge requests and S denotes those students that have knowledge points to learn. The teacher set will also be divided into two groups B and C. B contains those teachers not interested by current student S. In addition, A denotes allocated students while W denotes

1: $E = \cup_{j=1}^{n} Y_{s_j}, round = 0$
2: WHILE $E \neq \emptyset$
3: $D = \{s | s \in S, Y_s = \emptyset\}, S = S - D$
4: $A = \emptyset, B = \emptyset, C = \emptyset, W = \emptyset$
5: WHILE $T - B - C \neq \emptyset$
6: $t = argmax_{z \in T-B-C} |X_z|$
7: $p = argmax_{z \in X_t - W} |\{s | s \in S - A, z \in Y_s\}|$
8: IF $p == \emptyset$
9: $B = B \cup \{t\}$
10: ELSE
11: $V = \{s | s \in S - A, p \in Y_s\}$
12: $A = A \cup V, C = C \cup \{t\}, W = W \cup \{p\}$
13: $Y_z = Y_z - \{p\}, \forall z \in V$
14: END
15: END
16: $E = \cup_{j=1}^{n} Y_{s_j}$
17: $round = round + 1$
18: END
19: RETURN *round*

Fig. 3. Preferential-Teacher Algorithm

allocated knowledge points. Step 6 selects maximum capable teacher t. Step 7 selects knowledge point p of t interested by maximum students in $S - A$. If such p is not found, the teacher t joins into B. Otherwise, allocated students A, allocated teachers C, allocated knowledge points W, and knowledge point request Y_z are updated consequently. The inner loop stops when all teachers are grouped. Of course, E and *round* updated in order for next round of the inner loop.

4.3 Preferential-Knowledge Point Algorithm (PKPA)

The idea of third algorithm is in each iteration, first find knowledge point p_k with maximum unallocated interested students. Then select available teacher t_i that can serve p_k and has minimum corresponding capability.

In Figure 4, pending knowledge point set E and pending students S are initialized. The allocated students A, allocated teachers C, and allocated knowledge points W are set to empty. Step 6 selects the knowledge point p interested by maximum unallocated students. If p is not interested, that means no more knowledge points can be allocated. Otherwise, minimum capable teacher for knowledge point p is selected. If no available teacher exists, p joins into W. Otherwise, A, C, W, Y_z are modified accordingly. The inner loop stops when all knowledge points are inspected or no unallocated student exists. Within outer loop, $E, round$ are updated conveniently.

```
1: E = ∪_{j=1}^{n} Y_{s_j}, round = 0
2: WHILE E ≠ ∅
3:    D = {s|s ∈ S, Y_s = ∅}, S = S − D
4:    A = ∅, C = ∅, W = ∅
5:    WHILE P − W ≠ ∅
6:       p = argmax_{z∈P−W} |{s|s ∈ S − A, z ∈ Y_s}|
7:       V = {s|s ∈ S − A, p ∈ Y_s}
8:       IF V == ∅
9:          BREAK
10:      END
11:      t = argmin_{z∈T−C, p∈X_z} |X_z|
12:      IF t == ∅
13:         W = W ∪ {p}
14:      ELSE
15:         A = A ∪ V, C = C ∪ {t}, W = W ∪ {p}
16:         Y_z = Y_z − {p}, ∀z ∈ V
17:      END
18:   END
19:   E = ∪_{j=1}^{n} Y_{s_j}
20:   round = round + 1
21: END
22: RETURN round
```

Fig. 4. Preferential-Knowledge Point Algorithm

5 Performance Evaluation

We evaluate three heuristic algorithms running on a computer with 1.5 GHz CPU and 512 MB memory. The algorithms are implemented upon the *Matlab*. Scheduling meeting problem has six configuring parameters: l:number of knowledge point, m: number of teachers, n: number of students, f: size of $X_{t_i}, 1 \leq i \leq m$, g: size of $Y_{s_j}, 1 \leq j \leq n$, τ: cost tradeoff parameter. We relax τ in three proposed algorithms. In addition, each result is averaged over 10 runs through random sampling. We vary one configuring parameter while fixing others in order to reveal individual impact on algorithm performance. Each X_{t_i} and Y_{s_j} are randomly sampled on knowledge point set P. In addition, $\left(\bigcup_{j=1}^{n} Y_{s_j}\right) \subseteq (\bigcup_{i=1}^{m} X_{t_i})$ is ensured such that feasible solution exists.

l:number of knowledge point. In Figure 5, $m = 6, n = 40, f = 6, g = 5$. As number of knowledge point increases, the groups become less overlapping. Three algorithms need more rounds to complete the arrangement. Note that PKPA performs better than two others.

m:number of teachers. In Figure 6, $l = 20, n = 40, f = 6, g = 5$. Obviously, as the number of teachers grows up, more probably concurrent groups appear. Of course, the rounds derived by three algorithm decrease, although PKPA does the best.

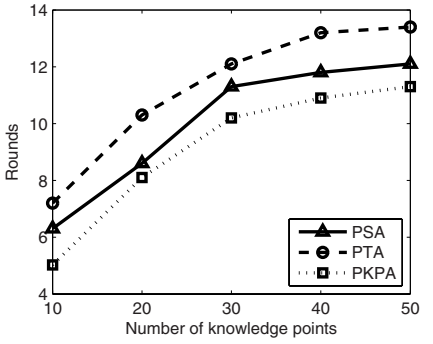


Fig. 5. Rounds vs. number of knowledge point

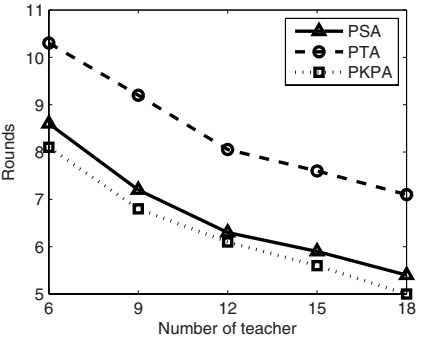


Fig. 6. Rounds vs. number of teacher

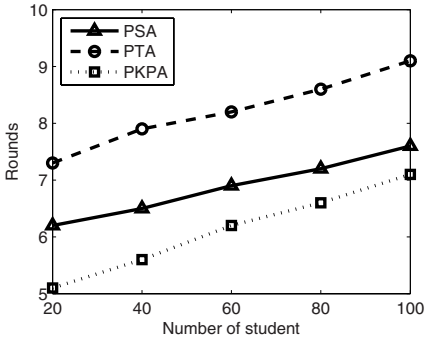


Fig. 7. Rounds vs. number of student

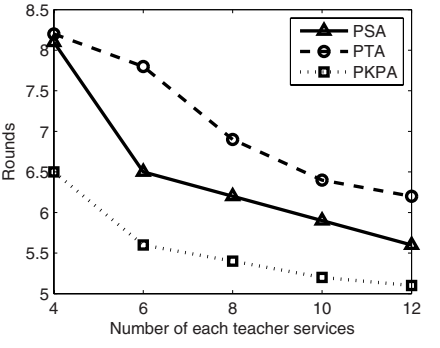


Fig. 8. Rounds vs. number of each teacher services

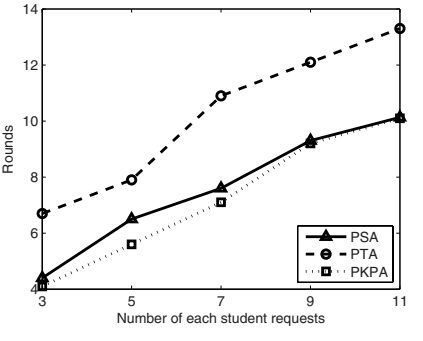


Fig. 9. Rounds vs. number of each student requests

n:number of students. In Figure 7, $l = 20, m = 11, f = 6, g = 5$. Increasing number of students means more requests need to be handled by teacher set. So the rounds increase definitely.

f:number of each teacher services. In Figure 8, $l = 20, m = 11, n = 40, g = 5$. Normally, teacher with larger service capability would decrease corresponding rounds since more probably the request can be satisfied.

g:number of each student requests. Figure 9 is similar to Figure 7. Here $l = 20, m = 11, n = 40, f = 12$. The number of the whole requests increases as g becomes large. Thus, rounds increases for three algorithms.

Summary: In five figures corresponding to each configuring parameter except for τ, the PKPA algorithm always outperforms two others. Potential reasons are: 1) selecting maximum student in each iteration. 2) choosing teacher with minimum capability would let other teacher can still contribute in following selections within same iteration.

6 Conclusion

This paper formulates scheduling meetings problem in distance learning scenario. Then three heuristic algorithms are proposed to quickly approximate the optimal solution. And the evaluation reveals that PKPA performs the best. In future work, one is to derive mathematical analysis of such formulation further, and the other is to incorporate τ into heuristic algorithm design.

References

1. Alvarez-Valdes, R., Parreno, F., Tamarit, J.M.: A Tabu Search Algorithm for Assigning Teachers to Courses. Top 10(2), 239–259 (2002)
2. Burke, E., Newall, J-P.: Multi-Stage Evolutionary algorithm for timetable problem. IEEE Transaction of Evolutionary Computer 3(1), 63–74 (1999)
3. Li, J.-n.: An Evolutionary algorithm for Solving Curriculum Schedule Problem of University. Microcomputer Development 13(10), 96–98 (2003)
4. Ueda, H., Ouchi, D., Takahashi, K., Miyahara, T.: A Co-evolving Timeslot/Room Assignment Genetic Algorithm Technique for University Timetabling. In: Proceedings of the 3rd International conference on Practice and Theory of Automated Timetabling, Konstanz, Germany, August,16-18, 2000, pp. 48–63 (2000)
5. Chu, S.-C., Chen, Y.-T., Ho, J.-H.: Timetable Scheduling Using Particle Swarm Optimization. In: Proceedings of the First International Conference on Innovative Computing, Information and Control, Beijing, China, August 30 - September 01, 2006, pp. 324–327 (2006)
6. Xie, F.-r.: A heuristic Numerical Algorithm for Solving the Problem of Curriculum Scheduling. Operations Research and Management Science 14(5), 36–40 (2005)
7. Rasmussen, R.V., Trick, M.A.: The Timetable Constrained Distance Minimization Problem. In: Proceedings of third International conference on CPAIOR, Cork, Ireland, May 31-June 2, 2006, pp. 167–181 (2006)
8. Wu, K., Havens, W.S.: Modeling an Academic Curriculum Plan as a Mixed-Initiative Constraint Satisfaction Problem. In: Proceedings of 18th Canadian Conference on Artificial Intelligence, Victoria, Canada, May 9-11, 2005, pp. 79–90 (2005)

Domain Level Page Sharing in Xen Virtual Machine Systems*

Myeongjae Jeon[1], Euiseong Seo[1], Junghyun Kim[2], and Joonwon Lee[1]

[1] CS Division, Korea Advanced Institute of Science and Technology
{mjjeon,ses,joon}@calab.kaist.ac.kr
[2] Samsung Electronics Co., Ltd.
junghyunx2.kim@samsung.com

Abstract. The memory size limits the scalability of virtual machine systems. There have been some researches about sharing identical pages among guest systems to reduce memory usage. However, they require memory overcommitment feature through swap mechanism which some virtual machines including Xen do not have. In this paper a new approach is proposed to share identical pages with designated sharing area. This approach reduces the memory usage as well as redundant I/O operations. Moreover, understanding the characteristics of certain shared pages becomes easier. The conceptional design was evaluated by simulation based on real-world applications.

Keywords: virtual machine, parallelism, memory management.

1 Introduction

The virtual machine systems, of which the heyday has been thought to be ended in 1970s, are now being resurrected due to the rapid improvement of hardware performance and storage capacity.

However, the scalability of virtual machines is limited by the hardware resources. The resources are able to be categorized into two groups; one is time sharing resources and the other one is space sharing ones. The shortage of time sharing resource is somewhat tolerable because it only induces some latency of processing time. The shortage of space sharing resources, however, such as memory and storage sometimes prohibits the addition of guest systems.

To overcome the scalability limitation from memory sharing, some approaches have been introduced following a way that a guest system shares existing identical pages which were allocated to other guest systems. However, these solutions are beneficial only with memory overcommitment feature. The overcommitment of memory in virtual machine systems means that the sum of the memory sizes that are allocated to guest systems is allowed to exceed the real hardware memory size. The most intuitive method to support memory overcommitment is

* This work was supported by KOSEF grant funded by the Korea government(MOST) (No. R01-2006-000-10724-0) and also partially funded by the MIC, Korea, under the ITRC support program supervised by the IITA.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 590–599, 2007.

using swap mechanism as in traditional operating systems. However, Xen [1] which is the representative of massive virtual machine system does not employ swap mechanism because of its design philosophy. Thus, the overcommitment of memory in Xen is restricted. As a result, the described solutions are not able to be directly adopted in Xen architecture.

This paper suggests page sharing scheme that the shared area is placed not in the guest operating systems but in the special designated area. This approach will aid the implementation of memory overcommitment in Xen because it helps to understand the property of shared contents well and to manage the size of shared memory area easily. There are permanent shared regions like operating system kernel and essential libraries. Thus, by checking the size of permanent shared area and shared counts, Xen can decide how much memory will be overcommitted safely.

The rest of this paper is organized as follows: Section 2 reviews existing research on page sharing among guest systems and Xen architecture also. Section 3 describes our approach for domain level page sharing scheme to support memory overcommitment in Xen. Section 4 presents the evaluation results. The new design concept for memory overcommitment without swap mechanism by using our approach is proposed in Section 5. Finally, section 6 summarizes our conclusions.

2 Back Ground

2.1 Xen Architecture

Guest systems in Xen are called *domains*. Xen virtual machine manager which is also called as *hypervisor* is located between hardware and domains and it distributes various resources to each domain. There is a special domain which is called *Domain-0*. It acts as control center that starts, stops and terminates other domains. The other domains which are ordinary guest systems are called *Domain-U*.

In virtual systems real memory can not be directly provided to each guest system because it may hinder the isolation among guest systems. Thus, Xen provides some real memory pages to each guest system as much as the virtual physical memory size of the corresponding guest system. To distinguish clearly, in terms of Xen, real memory is called *machine memory* and the virtual physical memory for each domain is called *pseudo-physical memory*. As in native environment, operating system in each domain distributes pseudo-physical memory to user-level tasks.

I/O requests from each guest system are actually handled in the Linux kernel located at Domain-0. Xen forwards I/O requests of all Domain-U to Domain-0. The forwarded requests will be handled with standard kernel functions in Domain-0. This model makes Xen have simple design and provide full standard and powerful interfaces to guest systems.

As a medium of delivering the I/O request, a structure called I/O ring lies in between Domain-0 and Domain-U. I/O ring consists of request and response

queues. Each queue is a circular queue and follows producer/consumer style. Domain-U produces I/O requests and put them to a queue in I/O ring. And then Domain-0 takes the requests at the other end of the queue and processes them. On the contrary, the processed results are put into I/O ring by Domain-0 and Domain-U gets the results for the requests they made from I/O ring.

Xeno Linux which is the modified Linux kernel for Xen employs front-end block device driver instead of standard block device driver to handles I/O requests for block devices such as hard disk and CD-ROM. Front-end block device driver puts the I/O requests of Domain-U into I/O ring. And it is responsible to get the results from I/O ring and to deliver them to the corresponding user level task in the domain.

Analogously Domain-0 requires a part to take request from I/O ring and to put the result into I/O ring. For this Domain-0 has back-end device driver. Back-end device driver gets a request from I/O ring and processes it with the existing kernel functions of Xeno Linux in Domain-0. After the processed result is queued in I/O ring, back-end device drivers a notification to the domain which made the request.

Xen itself can be light-weighted because I/O processing is done through existing operating system in Domain-0. However, it makes also Xen hardly have swap mechanism at machine memory level. Thus, the sum of memory which is allocated to all guest systems should be always less than machine memory capacity. Hot-plug memory and ballooning [2], with which guest systems give and take machine memory to and from other guest systems on demand, were proposed for more efficient memory use. It is, however, just a temporary solution because the actual allocated memory size of all guest systems is always less than that of machine memory after all.

2.2 Page Sharing Among Guest Systems

VMware ESX server[3] was a pioneer product that aimed to run massive guest systems. Accordingly, reducing memory consumption was directly related to its performance. VMware ESX server achieved much reduction of memory usage by *Content-based page sharing* scheme[4].

After a guest system gets a newly allocated page which is loaded with a certain disk block, Content-based page sharing compares the newly allocated and loaded machine page to all existing pages with MD5 hash function [5]. If there is a page with the identical content, then the mapping table in the guest system is modified so that the corresponding pseudo-physical page in the guest system points to the existing real hardware page instead of newly allocated one. This method needs not alteration of guest operating systems. It can be implemented with modification of only virtual machine manager.

However, executable files or data files stored in disks which are the primary target of Content-based page sharing do not change frequently. Thus, processing read requests for the already loaded pages to be shared is waste of resource. Moreover, a page which is located at the same disk block of a read-only filesystem has same contents naturally and does not require MD5 hash function to compare

the newly allocated page with the existing one of which target block is the same with that of the newly allocated one.

Although it was proposed before VMware ESX server, page sharing in Disco [6] is more efficient than VMware ESX in terms of processing redundant disk read requests. Disco captures DMA requests from guest systems and compares them to the recorded requests that were processed earlier. It becomes possible because Disco dominates all the hardware directly. If there is an existing page containing the requested block, the DMA operation for the request will not be actually started and the guest system which generates the request will share the existing page. This is fairly effective for not only memory saving but reducing read operations also. Intercepting DMA requests requires that virtual machine managers directly handle both hardware and I/O requests from guest systems by itself. Thus, this approach is hard to be straightly adopted in Xen, which handles I/O requests through the proxy guest system.

To achieve the improvement of scalability from all mentioned page sharing schemes, memory overcommitment should be supported by the virtual machine manager. Thus, they are not suitable to Xen which does not provide swapping of physical memory, which is an essential feature for the memory overcommitment, for the virtual systems.

3 Page Sharing for Xen

3.1 Design Overview

In most cases, Domain-U uses a small set of well-known operating systems such as Linux, FreeBSD and Microsoft Windows.In such environment many domains share read-only filesystems that contain operating system and frequently used program files and libraries. Each domain has their own writable filesystems for storing data and temporary files.

In this configuration, multiple pages scattered in different domains mostly happen to contain same disk block. The aim of this paper is to reduce memory usage and disk block read operations through the sharing of the multiple identical pages. This should be done with little overhead and fit to Xen architecture.

Before back-end block device driver processes it, our approach checks the read request whether there exists a machine page containing the target disk block or not. If it can not be found, the read request will be processed in the original path. On the contrary, if there exists the page in machine memory, the read request will not be processed and the page will be returned to front-end block device driver at Domain-U as the result for the request. Without distinction of the requesting domain, machine memory which was allocated to Domain-0 will be used as the page for the read request.

The suggested method quickly and precisely identifies page contents without requiring complex hash functions. It also reduces redundant read operations. The I/O handling in a virtual machine system is much slower than those in native systems [7]. Thus, the suggested scheme is expected to improve system performance as well as save memory usage.

3.2 Work Flow

The original processing flow of a disk read request in Xen is described as follows with Figure 1:

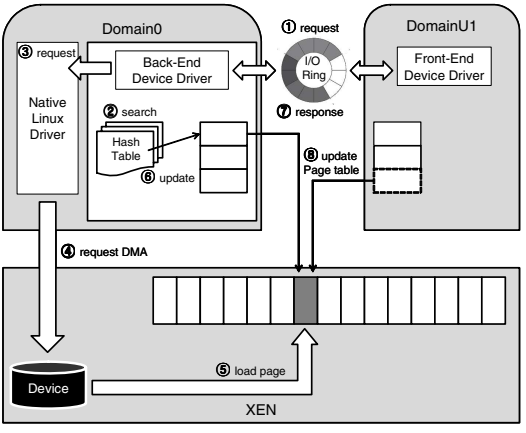


Fig. 1. Procedure for processing I/O requests in the suggested page sharing scheme

1. ①. The requester domain prepares an empty pseudo-physical page and gives access grant and address of the page to I/O ring. The I/O request is also put into I/O ring at this time.
2. ③. Back-end block device driver at Domain-0 maps the machine page mapped from the pseudo-physical page given from the requester domain to a pseudo-physical page in Domain-0. And it calls native Linux block device driver to load the target block into the mapped page.
3. ④. The native Linux driver initiates DMA operation for the request.
4. ⑤. DMA (in real hardware) reads the target block from disk and put it into the target page.
5. ⑦. After the finish notification from DMA, back-end block device driver delivers the result message to I/O ring and signal the requester that the operation is completed.

In our scheme the requesting domain does not give its machine address to Domain-0. On the contrary, Domain-0 provides its machine pages to the domain which requests block read from read-only filesystem. By manipulating mapping table the machine page which originally allocated to Domain-0 can be seen and used by the requester. Domain-0 can change the mapping states of machine pages to pseudo-physical pages by calling *hyper-call*. Hyper-call is function interfaces that connect Domain-0 and Xen hypervisor. In the suggested scheme, the flow of processing read request which is not in the memory is described as follows with Figure 1:

1. ①. ditto
2. ②. Domain-0 analyzes the read request. It searches block number table of existing pages. If it can not find a corresponding page, it allocates a new machine page for the request and maps the machine page address into a pseudo-physical page in Domain-0.
3. ③. ditto
4. ④. ditto
5. ⑤. ditto
6. ⑥. Upon completion of DMA operation, back-end block device driver add the device ID, block number and corresponding machine page address to the block number table.
7. ⑦. ditto
8. ⑧. Front-end block device driver maps the target pseudo-physical page address in ① to the machine page address.

In the suggested scheme, as ⑥ in Figure 1 a table to stores the device ID and block number of loaded pages is added and managed. In case there occurs a read request and the search result for the table returns a page that have the corresponding block, the processing of the read request is done only with manipulating corresponding mapping information. It is described as follows with Figure 1.

1. ①. ditto
2. ②. Domain-0 analyzes the read request. It searches block number table of existing pages. If it can find a corresponding page then it gets the machine address of that page from the table.
3. ⑦. ditto
4. ⑧. ditto

In the suggested scheme constructing of loaded block table, searching the table and allocating machine page for the newly loaded block are all done in Domain-0. Domain-0 is the best for the roles because Domain-0 actually processes all the I/O requests from all domains and controls all domains.

3.3 Data Structures

For the search of existing page contents we designed a data structure as illustrated in Figure 2. In Domain-0, there is a linked list of which element is each device ID. In other words, a device has an element in the list. A minor device [8] in Linux kernel is used for the unit of a device here. In a hard disk a minor device means not the disk itself but the partition in the disk. As a result, each of the elements corresponds to a mounted filesystem respectively. This is called device list.

A linked list is attached to each element of the device list. This is block list. An element in a block list has a block ID and also a machine page address of which a page currently containing the block. All the block elements in a certain device are gathered in the list which is attached to the device element in device list.

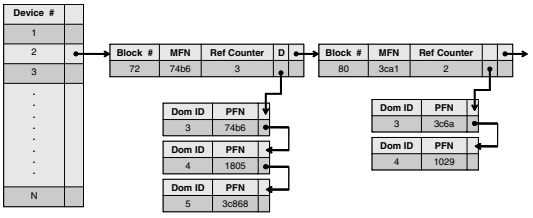


Fig. 2. Data structure for page sharing information

Each element of the block list also has a linked list respectively which is to store sharing information of the block. This is called allocation list. An element in this list has mapping information of a machine page, the pseudo-physical page which logically contains the block content and the domain which participates in the sharing of the machine page. If a domain which participates in the sharing frees a pseudo-physical page which references a machine page, the element for the domain will be removed from the allocation list. If all elements are removed, the corresponding machine page will be freed from Domain-0 also and it can be served for another disk block. A reference counter in an allocation list records the current number of sharing domains for a machine page.

Domain-0 works in with Xen hypervisor to allocate a new machine page, make a share of an existing machine page for other domains and deallocate an existing page without domains in use. On the contrary to data structure managed by Domain-0 in Figure 2, mapping tables are only accessible and writable in Xen hypervisor. Thus, Xen provides hyper-calls, which are analogous to system calls in traditional operating system, to Domain-0 for manipulating the mapping. The suggested method also uses these hyper-calls to manipulate the mapping tables. This approach makes actual sharing operations easier which are just modifying mapping tables.

4 Evaluation

Unfortunately, the suggested method has not been fully implemented yet since the structure of Xen related to the page management is rather complex and we lack of information on it. Thus, we evaluated the suggested scheme by simulation based on the real-world implementation of the described data structures and the functions. The simulation environment is described in Table 1. The target system was Xen 3.0 and Xeno Linux for it.

4.1 I/O Reduction and Page Sharing

Actually the amount of saved memory differs not from the sharing mechanism but from the property of workload and number of active domains. Thus, in this paper, the evaluation was not for comparison with existing page sharing schemes but for showing example benefits solely from the suggested page sharing.

Table 1. Evaluation environment

Virtual Machine Monitor	Xen 3.0
Guest OS	Linux Kernel 2.6.14 (Xeno Linux)
Memory for Sharing in Domain 0	128 MBytes
Memory for each Domain U	64 MBytes
Shared File System	executables and libraries

The workload for the evaluation is comprised of widely-used applications in the real-world. They are randomly started and finished in each domain. The applications for the evaluation are *Open-Office* (Officeware), *Firefox* (Web-browser), *Gimp* (Image editor) and *VLC* (Video player) and the executable file sizes of those applications are respectively 153.7 Mbytes, 27.2 Mbytes, 9.8 Mbytes and 9.1 Mbytes.

Each Domain-U has individual 64 Mbytes memory respectively. Domain-0 has 128 Mbytes memory. All the memory in Domain-0 except the portion for operating system and system programs is used for shared area.

The result was compared with the result under the original Xen system for each evaluation. The amount of shared pages is hard to be expressed easily because it changes continuously as time flows. Thus, the number of sharing which is also same as the number of reduced read operations is used for the comparison.

After an hour of execution, the results in Figure 3 were obtained. Each bar means the actual read requests processed in each environment.

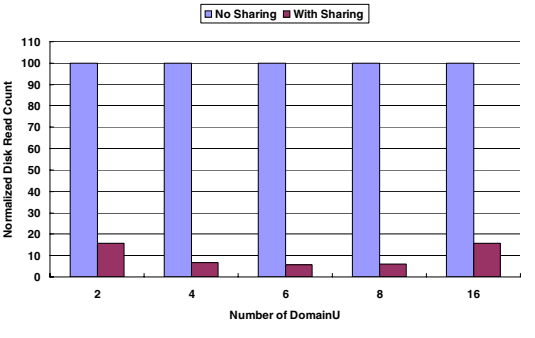


Fig. 3. Number of processed read requests under the suggested scheme

The use of shared area has the effects similar to buffer cache increment which prepares the frequent contents in advance by other domains. Each program was executed multiple times in the evaluation. Thus, the addition of shared area reduces much read requests. Comparing with the results under no sharing, only about 10% of total read requests are actually processed.

The ratio of actually processed requests tends to decrease as the number of domains in the sharing increases. This tendency, however, changes to the opposite

when there exist too many participating domains. With 16 domains, a little increase of the ratio happens. This is due to the fact that the working space exceeds the shared area when it contains many domains. Because each of them runs many applications concurrently and opens various files for the applications, the working set size for the shared filesystem increases as the number of running domains increases. Thus, adequate amount of shared area to the expected working set should be prepared for the best result.

4.2 Analysis of Overhead

To find an identical page in 128 Mbytes of shared area, we searched 32768 elements. If the search is done with well known B+ Tree, 32-order 4-depth tree is enough and the computational overhead for searching the tree is trivial.

To analyze the spatial overhead, we tracked the magnitude of it while the number of shared pages increases. The results show that the size of management structure grows linearly to the increase of shared area. For 128 MBytes of shared area shared with 4 domains, about 700 KBytes of memory is used for the management overhead. This is 0.005% of shared area capacity.

5 Concept of Non-Swap Overcommitment

Since Xen does not allow memory overcommitment, the actual improvement of scalability will be not gained, even if the implementation will be completed. To achieve the scalability improvement with the suggested method, Xen should have also memory overcommitment feature in it.

The most intuitive approach for memory overcommitment in virtual machine hypervisors is to employ swap mechanism. To support swap mechanism in virtual machine hypervisors, they should have device drivers for swap devices and filesystem drivers for swap filesystems. However, this policy does not fit the lightweight virtual machine hypervisors.

As a result, to increase the number of running domains without using swap mechanism, the feature is needed which allows the newly starting domain to use the surplus memory from the suggested page sharing method.

However, a blind overcommitment of the surplus memory may cause critical problems. When a pseudo-physical page which shares a machine page is freed, a new machine page should be mapped to the pseudo-physical page. If there is no available machine page due to the over-commitment at this time, the domain which owns the pseudo-physical page will be unable to proceed any more.

Thus, we propose a notion of *permanent share*. In read-only shared filesystem, there are files which should be always loaded in memory during the run-time such as operating system kernels and core libraries. Sharing for these files is named permanent share.

A permanent share is not being freed until the sharing domain is terminated. Thus, it is safe to provide surplus machine page, which is produced from permanent shares, to newly starting domains. This approach is expected to improve

the scalability of virtual machine systems significantly, because modern operating systems and core libraries have quite large executable file sizes.

Categorizing files with the permanent share property can be done transparently by analyzing the using patterns. However, when the categorizing fails, the domain with freed share may be unable to proceed its execution. Thus, accurate judgment of permanent share characteristics is a remaining issue.

6 Conclusion

Consolidating virtual servers into few virtual machine systems enables flexible configuration of cluster or distributed systems. The scalability of the server consolidation, however, is primarily limited by memory size.

This paper suggested a new page sharing design for Xen. It analyzes the block read requests that were forwarded from guest systems to controlling guest system and shares existing pages by modifying mapping tables when there exist identical pages. This scheme was verified with simulation using mock-up implementation.

As a further work, we also suggested a concept model of memory overcommitment without swap by using the suggested page sharing scheme based on permanent sharing notion where we are currently working on.

References

1. Barham, C.P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of vritualization. In: Proceedings of the 19th ACM symposium on Operating systems principles, pp. 164–177. ACM Press, New York (2003)
2. Schopp, J.H., Fraser, K., Silbermann, M.J.: Resizing memory with balloons and hotplug. In: Proceedings of the Linux Symposium, vol. 2, pp. 313–319 (2006)
3. Rosenblum, M.: Vmware's virtual platform: A virtual machine monitor for commodity pcs. In: Proceedings of the 11th Hotchips Conference (1999)
4. Waldspurger, C.A.: Memory resource management in vmware esx server. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementatioe (December 2002)
5. Rivest, R.L.: The MD5 message-digest algorithm. RFC 1321, MIT, RSA Data Security (April 1992)
6. Bugnion, E., Devine, S., Govil, K., Rosenblum, M.: Disco: Running commodity operating systems on scalable multiprocessors. ACM Transactions on Computer Systems 15(4), 412–447 (1997)
7. Cherkasova, L., Gardner, R.: Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In: Proceedings of USENIX 2005 Annual Technical Confernce, pp. 387–390 (2005)
8. Bovet, D.P., Cesati, M.: 13. In: Understanding the Linux Kernel. 3rd edn. O'Reilly, 536–537 (2006)

Parallel First-Order Dynamic Logic and Its Expressiveness and Axiomatization*

Zhiguo Zhang and Yunfei Jiang

Faculty of Information Science and Technology
Sun Yatsen University
Guangzhou City, 510275 China
lnszzg@mail.sysu.edu.cn

Abstract. For modeling the parallel actions, the quantified dynamic logic (QDL) is extended to Parallel First-order Dynamic Logic (PaFDL) with parallel action compositions. The composition is introduced as an operator $\cap$ on actions in the same syntax as in Peleg's CQDL but its semantics is defined differently from those of CQDL. The expressive power of PaFDL is proved to be the same as that of QDL. An axiomatic system is given and its first-order soundness and completeness are proved. Compared with other parallel or concurrent Dynamic Logics, PaFDL has a very easy and intuitive understanding for parallel actions as they are in the sequential models.

Keywords: Dynamic logic, First-order logic, Parallel actions, Expressiveness, Axiomatization.

1 Introduction

Propositional dynamic logic (PDL) was first proposed by Fischer and Ladner [FiL79] for describing the sequential program dynamic characteristics such as correctness, termination, and equivalence. It has received considerable attention, and many of its aspects have been thoroughly investigated [Nis79, Har84, HKT00]. Many investigations concern with complexity and axiomatization [Bal01, Dan84, KoP81, Lan05, LaL05, Lei81, Pra78]. PDL has been also extended to first-order level with many deep investigations [BeS01, GrS91, Har79]. Applications of dynamic logic to program verification and reasoning about actions and knowledge are also studied [PrS96, HRS87]. For modeling concurrent behaviors of multiagent systems, propositional dynamic logic has been extended to concurrent propositional dynamic logic (CPDL) and concurrent quantified dynamic logic (CQDL) by Peleg [Pel87a, Pel87b] with an extension of parallel actions. Peleg's approach views concurrency in its purest form as the dual notion of nondeterminism. Nondeterminism introduces splitting at a state into several branches, and letting the process choose between the different possible continuations. Analogously, concurrency means again splitting a node into several branches, but requiring the process to execute all possible

* This work is partially supported by Guangdong Key Laboratory of Information Security.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 600 – 607, 2007.

continuations. This is basically the classical concept of and/or decomposition, which occurs widely in logic, game theory, etc. It is shown that CPDL is strictly more expressive than PDL. A complete axiom system and its decidability for CPDL are provided. Some other investigations on concurrency of dynamic logic are presented in [Dan84].

In this paper, we introduce a parallel first-order dynamic logic (short for PaFDL) by adopting the syntax of Peleg's CQDL and give it a different semantics for parallel action compositions. The expressiveness of PaFDL is proved to be the same as QDL which is proposed by Harel et al. in [HKT00]. A sound and complete axiomatic system is provided for a restricted set of formulas of the form $A\rightarrow\langle\alpha\rangle B$. In the following, we give the syntactic definitions in Section 2, define their semantics in Section 3, and discuss the expressiveness in Section 4, the axiomatization in Section 5, and finally conclusion in Section 6.

2 Syntax

The syntax of parallel first-order dynamic logic (PaFDL) is based upon two kinds of symbols: *logical symbols* including the connectives $\neg$ and $\vee$, the punctuation marks (,), $\langle$, $\rangle$, [and], the equality symbol =, the existential qualifier $\exists$ and the universal qualifier symbol $\forall$, a countable set V of variables, the truth symbols *true* and *false*; *extralogical symbols* including a countable set P of predicate symbols, a countable set F of function symbols, and a countable set Π_0 of atomic action symbols. Each of function and predicate symbols has associated with it a natural number which is called its arity. 0-ary function symbols are called constants and 0-ary predicate symbols are called propositional constants. These countable sets constitute the basis for PaFDL. Complex formulas and complex programs over this basis are defined as follows.

Definition 1. (Basis) A basis for PaFDL is $B=(F,P,\Pi_0)$ of sets of symbols, where F, P and Π_0 are understood to be the sets of function symbols, predicate symbols, and action symbols respectively as described above.

Definition 2. (Terms) The set T_B of all terms of PaFDL over a basis $B=(F,P,\Pi_0)$ is inductively defined by:

(1) Every variable from V is a term. ($V\subseteq T_B$)
Every constant from F is a term.
(2) If $t_1, \ldots, t_n$ ($n\geq 1$) are terms and $f\in F$ is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is also a term.

Definition 3. (Formulas) The set Φ of all well-formed formulas of PaFDL over a basis $B=(F,P,\Pi_0)$ is inductively defined by:

(1) Every propositional constant from P is a formula. ($P\subseteq\Phi$)
The truth symbols *false* and *true* are formulas.
If t_1 and t_2 are terms, then $t_1=t_2$ is a formula.
If $t_1, \ldots, t_n$ ($n\geq 1$) are terms and $p\in P$ is an n-ary predicate symbol, then $p(t_1, \ldots, t_n)$ is also a formula.

(2) If A is a formula then ($\neg A$) ("not A") is a formula.
If A and B are formulas then ($A \vee B$) ("A or B") is a formula.
If A is a formula and x is a variable, then ($\exists xA$) and ($\forall xA$) are formulas.
If α is an action and A is a formula then $[\alpha]A$ ("every execution of α from the present state leads to a state where A is true") is a formula

Definition 4. (Actions) The set Π of all actions of PaFDL over a basis $B=(F,P,\Pi_0)$ is inductively defined by:

(1) Every atomic action is an action. ($\Pi_0 \subseteq \Pi$)
(2) If α and β are actions then ($\alpha;\beta$) ("do α followed by β") is an action.
If α and β are actions then ($\alpha \cup \beta$) ("do α or β, nondeterministically") is an action.
If α and β are actions then ($\alpha \cap \beta$) ("do α and β, in parallel") is an action.
If α is a action then α^* ("repeat α a finite, but nondeterministically determined, number of times") is an action.
If A is a formula then $A?$ ("proceed if A is true, else fail") is an action

The syntax for actions we adapted is exactly the same as Peleg's CQDL. However, we will have a different view of concurrency for parallel actions as described in the following sections.

3 Semantics

First we define a function patching operator as follows: if f: $D \to E$ is any function, $x \in D$ and $v \in E$, then $f[x/v]$: $D \to E$ is the function defined by

$$f[x/v](y) \stackrel{\text{def}}{=} \begin{cases} v & \text{if } x = y \\ f(y) & \text{otherwise} \end{cases}$$

We also need to define the domain *Bool* as

$$Bool = \{true, false\}.$$

As default, we always include this domain in our description.

Definition 5. (Interpretation) Let $B=(F,P,\Pi_0)$ be a basis for PaFDL. An interpretation of B is a pair $\mathcal{I}=(D,\mathcal{I}_0)$, where D is a non-empty set (called the domain or world of states of $\mathcal{I}$) and $\mathcal{I}_0$ is a mapping which assigns

(1) To every constant $c \in F$ an element $\mathcal{I}_0(c) \in D$;
(2) To every function symbol $f \in F$ of arity $n \geq 1$ a total function $\mathcal{I}_0(f)$: $D^n \to D$;
(3) To every propositional constant $a \in P$ an element $\mathcal{I}_0(a) \in Bool$, where *Bool* is the domain of truth values;
(4) To every predicate symbol $p \in P$ of arity $n \geq 1$ a predicate $\mathcal{I}_0(p)$: $D^n \to Bool$.

Definition 6. (Assignment) Let $B=(F,P,\Pi_0)$ be a basis for PaFDL and $\mathcal{I}=(D,\mathcal{I}_0)$ be an interpretation of B. A total function σ: $V \to D$ mapping variables to the domain D of $\mathcal{I}$ is called an assignment. In some context, an assignment is also called state. The set of all assignments for $\mathcal{I}$ is denoted by $\Sigma_{\mathcal{I}}$ or simply by Σ.

The definition of interpretation then can be extended to include:
(5) To every action symbol $\alpha \in \Pi_0$ a binary relation $\mathcal{I}_0(\alpha) \subseteq \Sigma \times \Sigma$.

An interpretation and an assignment together induce a mapping from every term to an element in the domain of the interpretation and from every formula to a truth value and from every action to a binary relation over assignments. It is clear that the interpretation must be extended inductively as follows to supply the intended meanings for the complex terms, actions and formulas:

Definition 7. (Semantics) Let $\mathcal{I}=(D,\mathcal{I}_0)$ be an interpretation of a basis $B=(F,P,\Pi_0)$ for PaFDL. To $\mathcal{I}$ is associated a functional, also denoted by $\mathcal{I}$, which maps every term $t\in T_B$ to a function $\mathcal{I}(t)$: $\Sigma\to D$ and every formula $A\in\Phi$ to a function $\mathcal{I}(A)$: $\Sigma\to Bool$ and every action $\alpha\in\Pi$ to a binary relation $\mathcal{I}(\alpha)\subseteq\Sigma\times\Sigma$. Each parts of this functional are defined inductively over T_B, Φ and Π as follows:

Semantics of terms

(1) If $c\in F$ is a constant, then $\mathcal{I}(c)(\sigma)=\mathcal{I}_0(c)$ for all assignments $\sigma\in\Sigma$.
If $x\in V$ is a variable, then $\mathcal{I}(x)(\sigma)=\sigma(x)$ for all assignments $\sigma\in\Sigma$.

(2) If $t_1, \ldots, t_n$ ($n\geq 1$) are terms and $f\in F$ is an n-ary function symbol, then $\mathcal{I}(f(t_1, \ldots, t_n))(\sigma)=\mathcal{I}_0(f)(\mathcal{I}(t_1)(\sigma),\ldots,\mathcal{I}(t_n)(\sigma))$ for all assignments $\sigma\in\Sigma$.

Semantics of actions

(1) For any atomic action a in Π_0, $\mathcal{I}(a)=\mathcal{I}_0(a)$.
(2) $(s,t)\in\mathcal{I}(\alpha;\beta)$ iff there exists a state z such that $(s,z)\in\mathcal{I}(\alpha)$ and $(z,t)\in\mathcal{I}(\beta)$.
(3) $(s,t)\in\mathcal{I}(\alpha\cup\beta)$ iff $(s,t)\in\mathcal{I}(\alpha)$ or $(s,t)\in\mathcal{I}(\beta)$.
(4) $(s,t)\in\mathcal{I}(\alpha\cap\beta)$ iff $(s,t)\in\mathcal{I}(\alpha;\beta)$ and $(s,t)\in\mathcal{I}(\beta;\alpha)$.
(5) $(s,t)\in\mathcal{I}(\alpha^*)$ iff there exists a non-negative integer n and there exist states $z_0, \ldots, z_n$ such that $z_0=s$, $z_n=t$ and for all $k=1..n$, $(z_{k-1}, z_k)\in\mathcal{I}(\alpha)$.
(6) $(s,t)\in\mathcal{I}(A?)$ iff $s=t$ and $\mathcal{I}(A)(t)=true$.

Semantics of formulas

(1) For any propositional constant a in P, then $\mathcal{I}(a)(s)=\mathcal{I}_0(a)$ for all s in Σ.
$\mathcal{I}(false)(s)=false$ and $\mathcal{I}(true)(s)=true$, for any s in Σ.
If t_1, t_2 are terms, then $\mathcal{I}(t_1=t_2)(s)=true$ if $\mathcal{I}(t_1)(s)=\mathcal{I}(t_2)(s)$, $\mathcal{I}(t_1=t_2)(s)=false$ if $\mathcal{I}(t_1)(s)\neq\mathcal{I}(t_2)(s)$, for all s in Σ.
If $t_1, \ldots, t_n$ ($n\geq 1$) are terms and $p\in P$ is an n-ary predicate symbol, then $\mathcal{I}(p(t_1, \ldots, t_n))(s)=\mathcal{I}_0(p)(\mathcal{I}(t_1)(s),\ldots,\mathcal{I}(t_n)(s))$ for all assignments $\sigma\in\Sigma$.

(2) $\mathcal{I}(\neg A)(s)=true$ if $\mathcal{I}(A)(s)=false$, $\mathcal{I}(\neg A)(s)=false$ if $\mathcal{I}(A)(s)=true$, for all s in Σ.
$\mathcal{I}(A\vee B)(s)=\mathcal{I}(A)(s)\vee\mathcal{I}(B)(s)$, for all s in Σ.
$\mathcal{I}([\alpha]A)(s)=true$ if s in $\{r$: for all states t, if $(r,t)\in\mathcal{I}(\alpha)$ then $\mathcal{I}(A)(t)=true\}$,
$\mathcal{I}([\alpha]A)(s)=false$ otherwise, for any s in Σ.
If $A\in T_B$ is a formula and $x\in V$ is a variable, then
$\mathcal{I}((\exists xA))(s)=$

$$\begin{cases} true & \text{if there is an element } d\in D \text{ of the domain such that } \mathcal{I}(A)(s[x/d])=true \\ false & \text{otherwise} \end{cases}$$

for all s in Σ.
If $A\in T_B$ is a formula and $x\in V$ is a variable, then

$$\mathcal{J}((\forall xA))(s)=\begin{cases} true & \text{if for all } d\in D \text{ of the domain such that } \mathcal{J}(A)(s[x/d])=true \\ false & \text{otherwise} \end{cases}$$

for all s in Σ.

We can define a new modal operator $\langle\rangle$ as follows:

$$\langle\alpha\rangle A =^{\text{def}} \neg[\alpha]\neg A.$$

Now consider a formula A. We shall say that A is valid in $\mathcal{J}$ or that $\mathcal{J}$ is a model of A, or "$\mathcal{J}\models A$", iff for all states s in $\Sigma_{\mathcal{J}}$, $\mathcal{J}(A)(s)=true$. A is said to be logically valid, or "$\models A$", iff for all interpretation $\mathcal{J}$, $\mathcal{J}\models A$. We shall say that A is satisfiable in $\mathcal{J}$ or that $\mathcal{J}$ satisfies A, or "$\mathcal{J}\backslash A$", iff there exists a state t such that $\mathcal{J}(A)(t)=true$. A is said to be logically satisfiable, or "$\backslash A$", iff there exists a model $\mathcal{J}$ such that $\mathcal{J}\backslash A$.

4 Expresiveness of PaFDL

We investigate the expressive power of PaFDL relative to quantified dynamic logic (QDL) with no $\cap$ operator. First we introduce a definition that allows us to compare different dynamic logics. If DL_1 and DL_2 are two different dynamic logics over the same basis, we say that DL_2 is as expressive as DL_1 and write $DL_1 \leq DL_2$ if for each formula A in DL_1 there is a formula B in DL_2 such that $\mathcal{J}(A\leftrightarrow B)(s)=true$ for all $\mathcal{J}$ and all s. Intuitively, $<$ and $\equiv$ mean "strictly less expressive than" and "of equal expressive power" respectively.

Lemma 1. QDL≤PaFDL.

Proof. This is directly from the syntactic definition of PaFDL. Actually, PaFDL is extended from QDL. □

Lemma 2. PaFDL≤QDL.

Proof. We should prove that for any A in PaFDL there is a formula B in QDL such that $\mathcal{J}(A\leftrightarrow B)(s)=true$ for all $\mathcal{J}$ and all s. Any formula A in PaFDL can be either containing operator $\cap$ or not. If A contains no operator $\cap$, then A is in QDL by the syntactic definition of PaFDL.

Now suppose that A contains the operator $\cap$. Without losing the generality and for simplification we consider only the case A is $[\alpha\cap\beta]B$. Given any interpretation $\mathcal{J}$ and $s\in\Sigma_{\mathcal{J}}$, by the semantic definition of PaFDL,

$$\mathcal{J}([\alpha\cap\beta]B)(s)=true$$
iff
$$s\in\{r\text{: for all states } t, \text{ if } (r,t)\in\mathcal{J}(\alpha\cap\beta) \text{ then } \mathcal{J}(B)(t)=true\}$$
iff
$$s\in\{r\text{: for all states } t, \text{ if } (r,t)\in\mathcal{J}(\alpha;\beta) \text{ and } (r,t)\in\mathcal{J}(\beta;\alpha) \text{ then } \mathcal{J}(B)(t)=true\}$$
iff
$$\mathcal{J}([\alpha;\beta]B)(s)=true \text{ and } \mathcal{J}([\beta;\alpha]B)(s)=true$$

iff

$\mathcal{I}([\alpha;\beta]B \wedge [\beta;\alpha]B)(s)=true$.

Clearly $[\alpha;\beta]B \wedge [\beta;\alpha]B$ is in QDL. □

Theorem 1. (Expressiveness) QDL≡PaFDL.

Proof. This is directly from the above two lemmas. □

This theorem shows that PaFDL has the same expressive power as QDL. Intuitively, PaFDL understands parallel actions in just the interleavable way as in QDL.

5 Axiomatization of PaFDL

Here we introduce an axiomatic system for the PaFDL calculus with an interpretation. Let $B=(F,P,\Pi_0)$ be a basis for PaFDL and $\mathcal{I}=(D,\mathcal{I}_0)$ be an interpretation of B. All semantically valid in $\mathcal{I}$ formulas of form $A \rightarrow \langle\alpha\rangle B$ are taken as axioms. This may lead some confusions with calculus because we consider that calculus has nothing to do with semantics. However, we usually can understand intuitively what the interpreted formulas to be true.

Axiom schemes

(A1) All instances of valid PaPDL formulas;
(A2) All instances of valid first-order formulas;
(A3) All formulas of form $A \rightarrow \langle\alpha\rangle B$ which satisfies
"for all s and t such that $(s,t) \in \mathcal{I}(\alpha)$, $\mathcal{I}(A)(s)=true$ implies $\mathcal{I}(B)(t)=true$",
where α is an atomic action.

Inference rule
(MP) modus ponens: from A, $A \rightarrow B$ infer B

If X is a set of formulas and A is a formula then we say that A is deducible from X in $\mathcal{I}$, or "$X \vdash_{\mathcal{I}} A$", if there exists a construction sequence $A_0, A_1, \ldots, A_n=A$ for A from the set of axioms and the inference rule (MP). Further, we say that A is deducible in $\mathcal{I}$ or "$\vdash_{\mathcal{I}} A$" iff $\emptyset \vdash_{\mathcal{I}} A$. X is said to be consistent in $\mathcal{I}$ iff not $X \vdash_{\mathcal{I}} false$.

Theorem 2. (First-order Soundness) For any PaFDL formula of the form $A \rightarrow \langle\alpha\rangle B$, for first-order A and B and action α containing first-order tests only,

$$\vdash_{\mathcal{I}} A \rightarrow \langle\alpha\rangle B \text{ implies } \mathcal{I} \models A \rightarrow \langle\alpha\rangle B$$

Proof. The proof of the soundness proceeds by induction on the compositions of actions. □

Theorem 3. (First-order Completeness) For any PaFDL formula of the form $A \rightarrow \langle\alpha\rangle B$, for first-order A and B and action α containing first-order tests only,

$$\mathcal{I} \models A \rightarrow \langle\alpha\rangle B \text{ implies } \vdash_{\mathcal{I}} A \rightarrow \langle\alpha\rangle B$$

Proof. The proof of the completeness is simplified as the following induction. For atomic α, it is clear that $\mathcal{J} \models A \rightarrow \langle \alpha \rangle B$ implies $\vdash_{\mathcal{J}} A \rightarrow \langle \alpha \rangle B$. For any composed action α = $\alpha_1 \cdot \alpha_2$, where operator "·" indicates one of ";", "∩","∪", we can prove the conclusion for α holds in the condition that the induction hypotheses for α_1 and α_2 both hold. And more for α* we can prove the same conclusion. □

6 Conclusion and Discussion

The Parallel First-order Dynamic Logic (PaFDL) is introduced with the syntactic and semantic definitions. The same syntax as Peleg's CQDL is adopted and semantics of PaFDL is defined differently from those of CQDL. The expressive power of PaFDL is proved to be the same as that of QDL. An axiomatic system is given and its first-order soundness and completeness are proved.

Compared with other parallel or concurrent Dynamic Logics, PaFDL has a very easy and intuitive understanding for parallel actions as they are in sequential models. According to Theorem 1, PaFDL has the same expressive power as its sequential counterpart QDL, not as the concurrent version CQDL. This indicates that PaFDL depicts parallel actions in a much like way the sequential models take.

Many other properties remain to be investigated including complexity, more extended axiomatization, and applications in reasoning about parallel actions and changes.

References

[BeS01] Beckert, B., Schlager, S.: A sequent calculus for first-order dynamic logic with trace modalities. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 626–641. Springer, Heidelberg (2001)

[Bal01] Balbiani, P.: A new proof of completeness for a relative modal logic with composition and intersection. Journal of Applied Non-Classical Logics 11, 269–280 (2001)

[Dan84] Danecki, R.: Nondeterministic propositional dynamic logic with intersection is decidable. In: Skowron, A. (ed.) Computation Theory. LNCS, vol. 208, pp. 34–53. Springer, Heidelberg (1985)

[FiL79] Fischer, M., Ladner, R.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences 18, 194–211 (1979)

[GrS91] Groenendijk, J., Stokhof, M.: Dynamic Predicate Logic. Linguistics and Philosophy 14(1), 31–100 (1991)

[Har84] Harel, D.: Dynamic logic. In: Gabbay, D., Guenthner, F. (eds.) (editors): Handbook of Philosophical Logic, vol. II, pp. 497–604. D. Reidel, Dordrecht (1984)

[Har79] Harel, D.: First-Order Dynamic Logic. LNCS, vol. 68. Springer, Heidelberg (1979)

[HKT00] Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (Massachusetts) (2000)

[HRS87] Heisel, M., Reif, W., Stephan, W.: Program verification using dynamic logic. In: Börger, E., Kleine Büning, H., Richter, M.M. (eds.) CSL 1987. LNCS, vol. 329, Springer, Heidelberg (1988)

[KoP81] Kozen, D., Parikh, R.: An elementary proof of the completeness of PDL. Theoretical Computer Science 14, 113–118 (1981)

[Lan05] Lange, M.: A lower complexity bound for propositional dynamic logic with intersection. In: Schmidt, R., Pratt-Hartmann, I., Reynolds, M., Wansing, H. (eds.) Advances in Modal Logic, vol. 5, pp. 133–147. King's College Publications, London (2005)

[LaL05] Lange, M., Lutz, C.: 2-EXPTIME lower bounds for propositional dynamic logics with intersection. Journal of Symbolic Logic 70, 1072–1086 (2005)

[Lei81] Leivant, D.: Proof Theoretic Methodology for Propositional Dynamic Logic. In: Díaz, J., Ramos, I. (eds.) Formalization of Programming Concepts. LNCS, vol. 107, Springer, Heidelberg (1981)

[Nis79] Nishimura, H.: Sequential method in propositional dynamic logic. Acta Informatica 12, 377–400 (1979)

[Pel87a] Peleg, D.: Concurrent dynamic logic. Journal of the ACM 34, 450–479 (1987)

[Pel87b] David, P.: Communication in Concurrent Dynamic Logic. J. Comput. Syst. Sci. 35, 23–58 (1987)

[Pra80] Pratt, V.: A near-optimal method for reasoning about action. Journal of Computer and System Sciences 20, 231–254 (1980)

[Pra78] Pratt, V.: A practical decision method for propositional dynamic logic. In: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, pp. 326–337. ACM Press, New York (1978)

[PrS96] Prendinger, H., Schurz, G.: Reasoning about action and change: A dynamic logic approach. Journal of Logic, Language, and Information 5(2), 209–245 (1996)

Efficient Voice User Interface System Using VoiceXML and ASP.NET 2.0

Byung-Seok Kang[1] and Gi-Jong Yoo[2]

[1] Department of Electronics Engineering, Korea University
1, 5-ga, Anam-dong, Sungbuk-gu, 136-701, Seoul, Korea
kbsgasu@korea.ac.kr
[2] Graduate School of Education, Ajou University
San 5, Wonchon-dong, Yeongtong-gu, 443-749, Suwon, Kyonggi-do, Korea
mathink@naver.com

Abstract. The web-based application by VoiceXML service on the Internet is gradually being accepted for the human-machine interaction because it provides the speech-enabled function and makes telephone access a reality. Many companies are interested in building the effective dynamic Voice User Interface (VUI) system into the architecture of the already existing web application. The previous papers [1, 2] suggest that they demonstrate how to design and implement using VoiceXML and Active Server Pages. However, they have used only one server script language, so it is not efficient. For that reason, we have built another design that is more efficient for VoiceXML. Experimental results demonstrate that ASP.NET 2.0 shows the highest communication success rate and the lowest response time for web surfing.

1 Introduction

Today the computer distinguishes itself as a key player in the everyday human activity, be it business, research, engineering, or entertainment. The invention of the World Wide Web gave the computer even greater importance. By bringing the entire globe under its orb, the World Wide Web opened before us a new world characterized by the use of computer applications in every field. Voice and web technologies assumed a definite shape in the last decade of the 20th century and soon sparked a series of unprecedented changes in the way people interact long distance. There are a host of competent technologies, such as VoiceXML 2.0, that facilitate fast and accurate transfer of information all over the world. Furthermore, the Microsoft new computer language Active Server Page.NET 2.0 is powerful for designing the VUI system.

Many companies and personal users are using VXML for their customer web service. D. Mecanovic proposes the Voice User Interface Design for a telephone application [1]. They proved that the long and descriptive prompts make navigation difficult and female text-to-speech (TTS) voice is preferred in dynamic VUI. R. Vankayala and H. Shi [2], in their paper "Dynamic Voice Use Interface using VXML and Active Server Pages" implemented and demonstrated an existing e-Commerce

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 608 – 616, 2007.

web application by using the BeVocal.com server and Microsoft IIS Web server. Furthermore M. Tsai [5] deploys a web-based Mandarin dialogue system, in which a user can use either a telephone channel or VoIP by personal computer to access the voice server. But no one suggested which programming language is the most efficient when it is used with VXML for existing web application services. In this paper, we perform comprehensive experiments to find out the best implementation tool to build an efficient VUI system. Specifically, we consider four tools: ASP, ASP.NET 2.0, JSP, and PHP. Experimental results demonstrate that ASP.NET 2.0 has the highest communication success rate and the lowest response time. This paper is organized as follows. Section 2 discusses some background information. Next, the design utilizing VoiceXML with some famous web programming languages is presented in section 3. Simulation results are presented in Section 4 to show that our proposed design can provide a good performance in the communication success rate and reduce the response time. Finally, Section 5 concludes the paper.

2 Background

2.1 VoiceXML System

While HTML assumes a graphical web browser with display, keyboard, and mouse, VoiceXML assumes a voice browser with audio output, audio input, and keypad input. Audio input is handled by the voice browser's speech recognizer. Audio output consists both of recordings and speech synthesized by the voice browser's text-to-speech system.

A voice browser typically runs on a specialized voice gateway node that is connected both to the Internet and to the public switched telephone network (see Figure 1). The voice gateway can support hundreds or thousands of simultaneous callers, and can be accessed by any one of the world's estimated 1,500,000,000 phones, ranging from antique black candlestick phones up to the very latest mobiles.

The user interacts with a Web site over the phone using a VoiceXML Browser, which is hosted on a Gateway. Instead of rendering and interpreting HTML, the VoiceXML Browser renders and interprets VoiceXML. The Gateway is the key

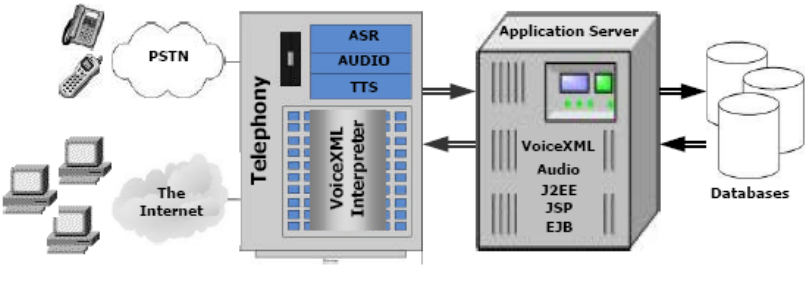


Fig. 1. VoiceXML serving architecture

bridge technology, responsible for VoiceXML Browser, ASR Resource, TTS Resource, Telephony Resource, Audio Resource and TCP/IP Resource.

2.2 ASP.NET 2.0

Active Server Pages.NET (ASP.NET) is a web development technology from Microsoft. Part of the .NET Framework, ASP.NET allows developers to build dynamic web applications and web services using compiled languages like VB.NET and C#. Using Visual Studio, the development tool from Microsoft, web developers can develop very compelling applications using ASP.NET, with the ease of drag-and-drop server controls. Currently in its next major release, ASP.NET 2.0 is slated to be released in November 2005.

ASP.NET 2.0 is a compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET 2.0 can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box.

Two aspects of ASP.Net 2.0 makes it fast (compiled code and caching). In the past, the code was interpreted into "machine language" when website visitor viewed web page. Now, with ASP.Net 2.0 the code is compiled into "machine language" before visitor ever comes to web site. Caching is the storage of information that will be reused in a memory location for faster access in the future. ASP.Net 2.0 allows programmers to set up pages or areas of pages that are commonly reused to be cached for a set period of time to improve the performance of web applications. In addition, ASP.Net 2.0 allows the caching of data from a database so the website isn't slowed down by frequent visits to a database when the data doesn't change very often.

3 Design and Implementation Using VoiceXML and ASP.NET 2.0

This section gives the main implementation details of our site. We describe our algorithm and four designed systems.

3.1 System Architecture

To build VUI systems, we use VoiceXML café, “BeVocal.com” [6] for a VoiceXML server. The BeVocal Café is a world-class, web-based development environment that provides all the tools and resources developers need to create their own innovative speech applications.

In our VUI systems, two web servers (Microsoft IIS and Apache web server), two operating systems (Windows 2003 server and Linux server), four web programming languages (ASP, ASP.NET 2.0, JSP, and PHP), and two database systems (MS-SQL 2005 and MySQL 5.0) are used. Figure 2 shows the VUI system architecture.

3.2 Algorithm of the Web Surfing System

To demonstrate the behavior of VUI systems, the following application scenario is adopted. Once a user calls a VUI homepage, the VUI system says “Welcome to my homepage. Which do you want? Notice, Free board, Public data or On-line poll?” If

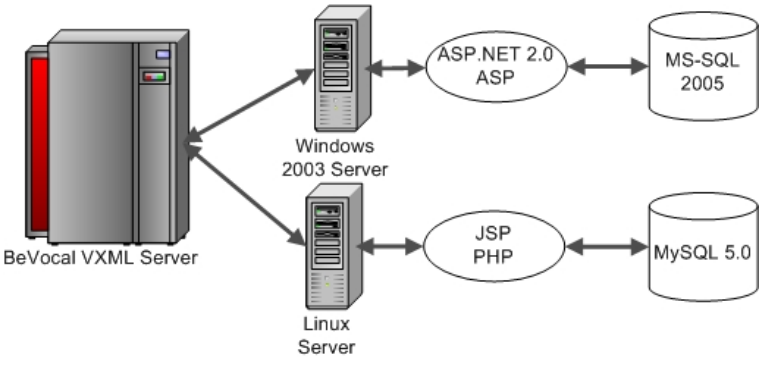


Fig. 2. VUI architecture

the user selects a free board (b), the system says "Do you want to write or not?". If the user chooses "yes", the system shows a "write article" page; otherwise, the system asks you a next question. Detailed application scenario is illustrated in Figure 3.

The source code which is below is the detail for main page in Figure 3's application scenario [7]. During some system design we describe a part of source code under ASP.NET 2.0.

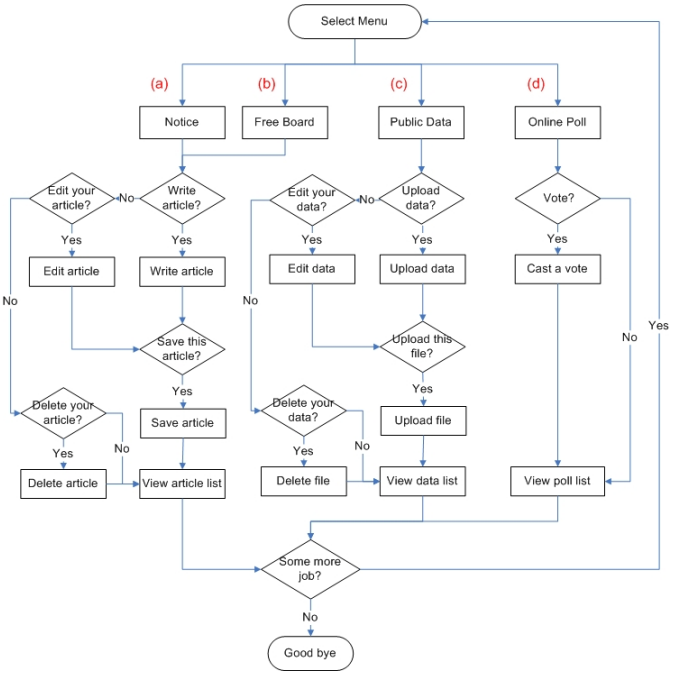


Fig. 3. Application scenario in VUI systems

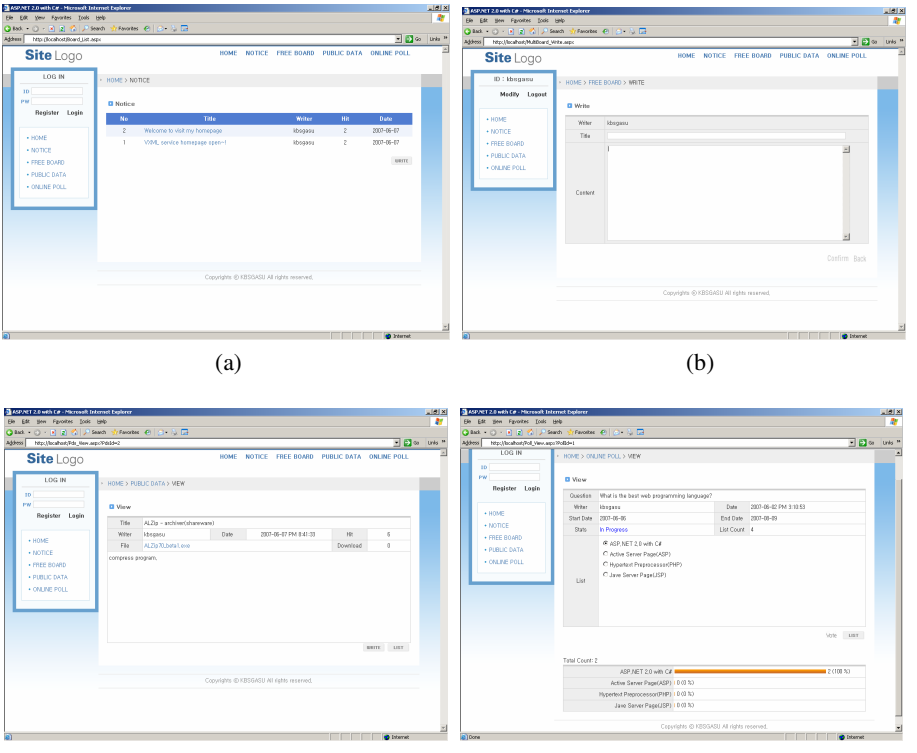
Sample Source Code of Main Page

```
<%@ Page Language="C#" MasterPageFile="~/Default.master"
Title="Simulation no.4 - ASP.NET 2.0 with C#" %>

<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form id="main" scope="dialog">
 <prompt bargein="true">
  Welcome to visit my homepage. Which do you want?
 </prompt>
 <grammar> Notice | Free Board | Public Data | Survey </grammar>
 <noinput>
  No response, say one more please.
  <reprompt/>
 </noinput>
 <noinput count="4">
  Disconnect the phone.
 <disconnect/>
 </noinput>
 <nomatch>
  Say one more please.
 <reprompt/>
 </nomatch>
 <filled namelist="user_input" mode="all">
 <if cond="user_input == 'Notice'">
  <goto next="/List.aspx?TblName=Notice" fetchhint="safe"/>
 <elseif cond="user_input == 'Free Board'"/>
  <goto next="/List.aspx?TblName=Notice" fetchhint="safe"/>
 <elseif cond="user_input == 'Public Data'"/>
  <goto next="/List.aspx?TblName=Notice" fetchhint="safe"/>
 <elseif cond="user_input == 'Survey'"/>
  <goto next="/Poll_List.aspx" fetchhint="safe"/>
 <else/>
  <disconnect/>
 </if>
 </filled>
</form>
</VXML>
```

If users request the web service through voice interface, the BeVocal server should interpret the voice and connect with this site for response. Then the web server shows the correct web page and waits for the next request. Figure 4 shows some sample web pages.

First, in figure 4(*a*), visitors view a notice board. Second, Figure 4(*b*) shows a guest write the free board. Third, figure 4(*c*), users connect to the public data for download some useful data. Finally in figure 4(*d*), users view the result of online poll page.



(a) (b)

(c) (d)

Fig. 4. (*a*) View notice list. (*b*) Write the free board. (*c*) Download some data. (*d*) The result of online poll.

4 Simulations

In the implemented VUI systems, we measure the average response time and the success rate. The average response time is the elapsed time to obtain the result page when a user requests a page, and the success rate is the probability that a correct page is obtained. Figure 5, 6 shows the average response time as the number of web pages surfed by users using VUI system or not. As you can assume, VUI is much more efficient than usual web application system in using circumstances (figure6). It is shown that ASP.NET 2.0 has the shortest response time and JSP and ASP exhibit longer response time than PHP and ASP.NET 2.0. This can be explained by two reasons: 1) JSP, ASP, and PHP interpret common language runtime codes whenever a user requests a page. On the other hand, ASP.NET 2.0 pre-compiles common language runtime codes before the user visits the web site, and thus it can display the result page immediately without any interpretation; 2) ASP.NET 2.0 employs a caching scheme to allocate frequently used code and data in main memory when a user visits a web site.

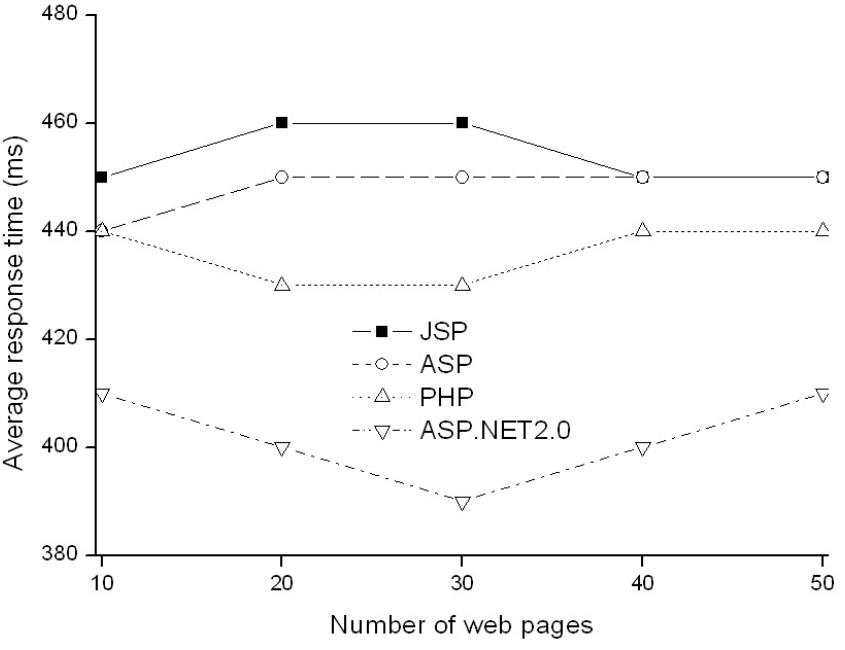


Fig. 5. Average response time not using VUI system

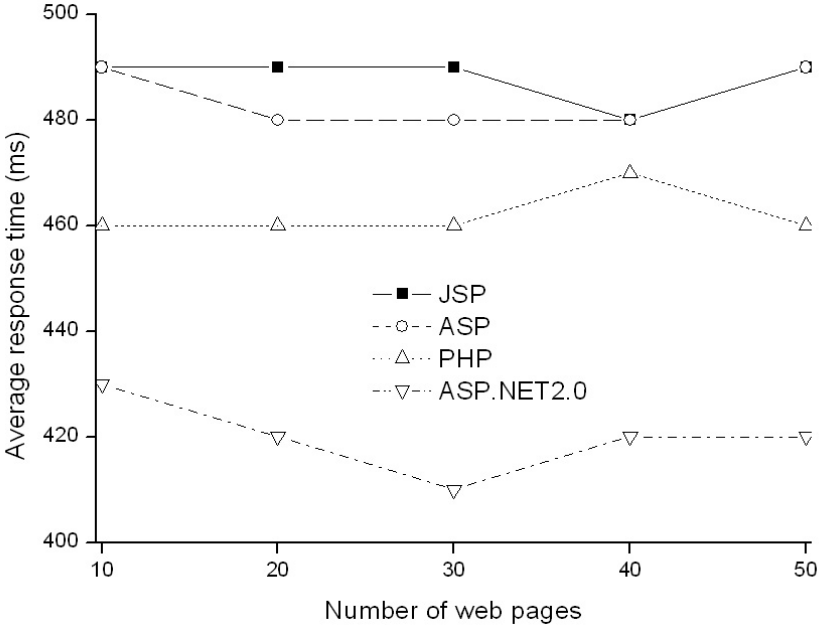


Fig. 6. Average response time using VUI system

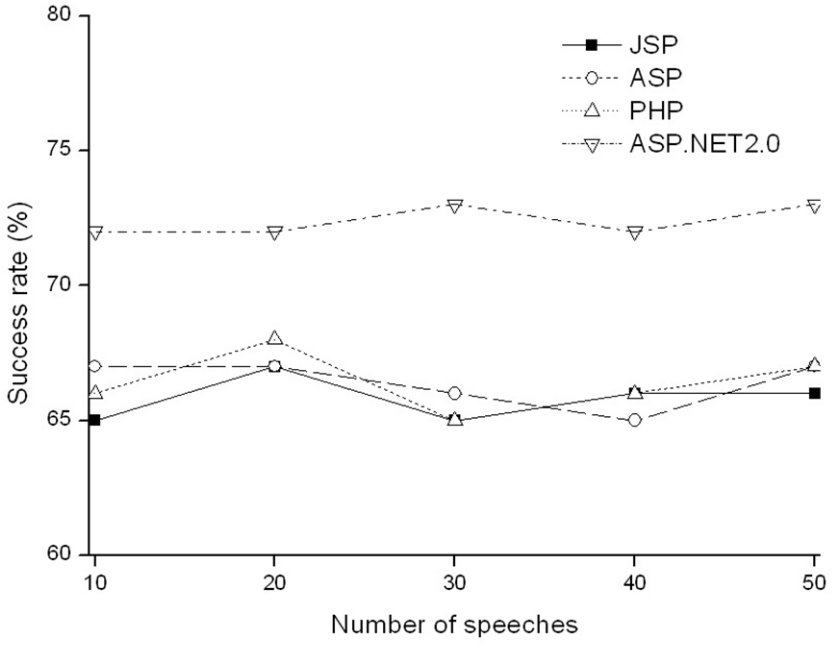


Fig. 7. Success rate

Figure 7 show the success rate as the number of speeches. It can be found that ASP.NET 2.0 has the highest success rate compared with other tools. All of these experimental results demonstrate that ASP.NET 2.0 is the most suitable programming language in VUI system.

5 Conclusion and Future work

This paper describes which web programming language reduces the response time with VXML. We design four other VXML systems. We also propose the guideline to use the VUI service without changing previous infra structure. Experimental results indicate that ASP.NET 2.0 can significantly reduce the average response time and provide higher success rate, compared with other tools, i.e., JSP, PHP, and ASP. In our future work, we will investigate VUI systems using VoiceXML 3.0 which is a new release from the W3C's voice browser working group [4].

References

1. Mecanovic, D., Shi, H.: Voice User Interface Design for a Telephone Application using VoiceXML. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 1058–1061. Springer, Heidelberg (2005)
2. Vankayala, R., Shi, H.: Dynamic Voice User Interface Using VoiceXML and Active Server Pages. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 16–18. Springer, Heidelberg (2006)

3. VoiceXML Forum (June 2007), http://www.voicexml.org
4. W3C Voice Brower Activity (June 2007), http://www.w3c.org/Voice
5. Tsai, M.-j.: The VoiceXML dialog system for the e-commerce ordering service. In: Proc. of the Ninth International Conference, May 24-26, 2005, vol. 1, pp. 95–100 (2005)
6. BeVocal Café, VoiceXML development environment (June 2007), http://www.bevocal.com
7. XML Web Services Created Using ASP.NET and XML Web Service Clients (June 2007), http://msdn2.microsoft.com/en-us/library/7bkzywba.aspx

Array Modeling in Java Virtual Machine

Wu Weimin[1], Li Kailun[2], and Su Qing[3]

[1] Computer Faculty, Guangdong University of Technology, Guangzhou 510006, China
[2] Guangzhou Branch, People's Bank of China, Guangzhou 510050, China

Abstract. Array is an important feature in Java and Java Virtual Machine. In spite of its importance, it has not been modeled by any existing Java Virtual Machine Models. In this paper, we define an extending model which uses an existing model as a basis and give the hierarchy of these two models, to model the array. In the extending model, we model the array in three steps. The first step is adding array related instructions in a formal way. The second step is refining the type compatibility to include array types. The last step is implementing array loading process also in a formal way. In the last part of thesis, we give the future work of extending other important features in Java and Java Virtual Machine.

1 Introduction

Java is an object-oriented programming language with a widespread use, and the java compiler translates Java source code to bytecode, which executes on the Java Virtual Machine (JVM).[1]

Compared to other object-oriented languages, there are many distinct features in Java, and array is one of them. And array is also an important feature in Java and JVM. The distinction and importance of it are as follows. First, in Java programming language, array is a most frequently used data structure to contain elements. Second, in Java type system, array type is a kind of reference type (the other two are class type and interface type). Third, in object creating, other then any other kinds of objects, array is a full-fledged object and is dynamically created, and it generally includes basic type array and object reference array. Fourth, in object loading, array loading is different from class or interface loading. Last but not least, in JVM instruction set, the JVM uses special bytecode to handle array. [2]

For this distinct and important feature in Java and JVM, it is significant to describe it in essence to help us in designing, programming, etc. The best and precise way to describe the essence is to build a model in a mathematical way. And because the JVM involves type, object creating and loading, and instruction set, so the most suitable model to build on is JVM model. After building the model, we can take the advantages as follows. First, it can help us precisely find the compiling error or runtime error which is directly or indirectly caused by misusing of array. Second, it can help us know what happens to the array behind the scene in a mathematical way, and thus help us design more robust programs.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 617–623, 2007.

So far, several formalizations of the JVM model have been proposed. However, they provide only insight into one or few aspects of the machine, not the whole machine, and the array is not modeled in any of these models. So it means we need to extend an existed model to support array. The most rigorous and comprehensive one among these model is the machine proposed by Egon Borger and Wolfram Schulte (for clarity, we call it BSM, namely, Borger and Schulte Model). [3]. This machine can be validated and verified by standard techniques because it is defined by Abstract State Machines (ASM), which have a simple but precise semantic foundation. [4]

In BSM, the model can be described as a hierarchy of four submachines. Fig. 1. shows the hierarchy. The basic stack machine VMI supports instructions which are used for compilation of imperative programs. Typical instructions are: load and store a variable, apply arithmetic and relational operators, and jump. VMI is upgraded to VMC by including instructions which are used for the compilation of Java static features, such as class fields, class methods and class initializers. VMC can be extended to VMO which supports instructions for Java object-oriented features, such as instance creation, instance field access, instance calls with early or lately binding and type casts. And the topmost machine VME provides instructions with respect to exception [3]. And this structural decomposition is based on the orthogonality of various language features of Java. [5,6]

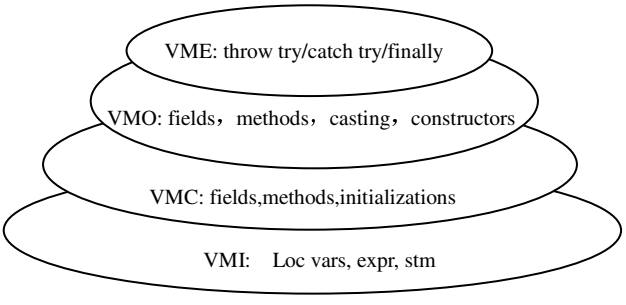


Fig. 1. Structural Decomposition of BSM

2 The Model of Extending BSM

Based on the current situation of JVM and the importance of array in JVM, we extend the BSM to support the function of array. And we call this extending model the Extending BSM (short for EBSM).

We add a VMA machine which supports array on top of the topmost model VME to extend the BSM. The main reason is as follows: when we extend the machine to support array, we should refine some functions and add some functions. If we separate these functions in four levels, then the description of the functions will not be centralized and the difference between BSM and EBSM will not be clear enough. And if we create a new level above the BSM, then these two problems will be solved. Fig. 2. shows the structural decomposition of EBSM.

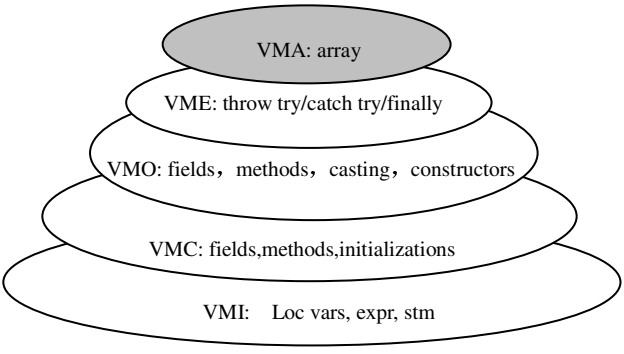


Fig. 2. Structural Decomposition of EBSM

3 VMA Modeling

We implement the VMA machine by three steps. First, we add 20 instructions which involving array into the instruction set of the JVM. Second, we refine the type compatibility to include the array type. Third, we refine the loading method to add the process of loading and linking array.

3.1 Adding Array Instructions

There are about 20 array related instructions in JVM. They can be divided into three kinds. The first kind is loading and storing array elements, contains aaload, aastore, baload, bastore, caload, castore, daload, dastore, faload, fastore, iaload, iastore, laload, lastore, saload, sastore. The most difference among these instructions are the type of the operand. Details of these instructions can be seen in [2]. The second kind is creating array, contains anewarray, newarray, multianewarray, which mean creating array of reference type, creating array of basic type and creating multiple array respectively. The third kind is getting length of array, which has only one instruction: arraylength. The first and the second kind are more important than the third kind, so we describe these two kind instructions in detail.

3.1.1 Instructions of Loading and Storing Array Elements

Because the execVM part of the BSM, which defines the process of instruction executing, uses the free data type to abstract the difference between non-array type, including reference type and basic type, so the first kind of instructions can be abstract to loadarrayelem and storearrayelem, which mean loading array element and storing array element respectively. Prog. 1. shows this kind of instructions.

Prog. 1. Instructions of loading and storing array elements

```
execVM (redef)==…
loadarrayelem()•
if newopd•wr•wi=opd• #wr =r• #wi=i  then
newopd(#newopd+1):=r[i]
```

```
pc:=pc+1
storarrayelem()•
if newopd•wr•wi•wv=opd• #wr=r• #wi=i• #wv=v then
r[i]:=v
pc:=pc+1
```

3.1.2 Instructions of Creating Array

To implement the instructions of creating array, we should first define a type called AState to denote the state of an array. Prog. 2. shows the AState type and its related State and InitialState.

Prog. 2. AState Type

```
Type                AState:= NotInited | Inited
State               aState:ANm • AState
Initial State       aState(c)=NotInited
```

To implement the instruction newarray, we define five steps. First, we get the array type according to the basic type in parameter using the function arraytype. Second, if the class of the array is not already loaded, we should first load the class of the array type using callLink function. Third, we set the class and dimension of the new created reference using function aOf and countOf respectively. Fourth, we initialize the array using the default values. Last, we update the operand and the PC register. The implementation of instruction anewarray is similar to the newarray. For clarity, we do not describe it in detail.

The most distinct differences between newarray and multianewarray is the latter uses a function initArrayElem to init the elements of the multiple array using function elemType which returns the element type of the array argument. The element may also be an array, so this call may be recursive. Prog. 3. shows the instructions of newarray and multianewarray.

Prog. 3. Instructions of newarray and multianewarray

```
execVM (redef)==…
newarray(t) •
at := arraytype(t)
if newopd •wc = opd • # wc =c then
if aState(at) := NotInited
    callLink(cLd(meth), cNm(f))
aOf(r) := at
countOf(r) := c
for all e in afield(at)
elem(r ,e) := default(e)
```

```
newopd :=new opd • [r]
pc := pc +1
where r = new (dom(aOf))

multianewarray(t, d) •
at := multiarraytype(t, d)
if newopd •wc1…•wcd = opd • # wc1 =c1 •… # wcd =cd then
if aState(at) := NotInited
    callLink(cLd(meth), cNm(f))
aOf(r) := at
i := 1
countOf(r) := ci
for all e in afield(at)
elem(r ,e) :=(i = d) ?default(e): initArrayElem
(elemtype(at),ci+1 )
newopd :=newopd • [r]
pc := pc +1
where r = new (dom(aOf))

initArrayElem(at, ci)
aOf(r) := at
countOf(r) := ci
for all e in afield(at)
elem(r ,e) := (i = d) ?default(e)
:initArrayElem(elemtype(at),ci+1 )
where r = new (dom(aOf))
```

3.2 Type Compatibility of Array Type

Because the propagateVM part (which defines the process of verifying byte code) and the execVM part of the BSM involves type compatibility, so we refine the function compat and the operator '≤'to include type compatibility of array.

Prog. 4. shows the refined function and operator. According to the JVM specification[2], it is compatible when two arrays are of the same dimension and the element types of the two arrays are type compatible. In model, the function isArray returns true if parameter is actually an array, and the function dim returns the dimension of the array argument.

Prog. 4. Type compatibility of array type

```
isArray(C1) • isArray(C2) • dim (C1) = dim (C2) •elemType
(C1) • elemType(C2)
• Compat(C1, C2) = true
isArray(C1) • isArray(C2) • dim(C1) = dim(C2) •elemType
(C1) • elemType(C2)
• C1 • C2
```

3.3 Array Loading

For array loading, if the element type of the array is a reference type, then according to the JVM specification[2], JVM first loads the element type (may lead to recursively loading), and then adds the array type to the name space of the environment. If the element type of the array is a basic type, then just adds the array type to the name space of the environment.

Because no matter what classLoaders (system or user-defined) are defined, the function of loading and linking are eventually found in findSystemClass, defineClass and resolveClass in class called classClassLoader, so we refine the InvInstance method taken each of these three method names as argument in the execVM of BSM for array loading. Prog. 5. shows the refining rule for findSystemClass. We refine similarly the execution rules for the other two methods. For clarity, we do not show it in Prog. 5. The black part of the program shows what we have refined.

Prog. 5. Instructions of array loading

```
execVM (redef)==…
InvInstance(bind, findSystemClass) •
if cinitd(cNm(findSystenClass)) •newopd •[ld, cn] = opd
•ld <> null then
  let c = (sysLd, cn) in
if unloaded(c) then
     if ¬ isArray(c) then
        loadVM(c)
     else
        bc := elementType (c)
        if(isReferenceType(Class(bc))) then
            loadVM(bc)
            addArrayToEnv(c, lc)
        else
            addArrayToEnv(c, lc)
else if ¬ cinitd(c) then
```

```
      if ¬ isArray(c) then
linkVM(c)
      else
           bc := elementType (c)
         if(isReferenceType(Class(bc))) then
              linkVM(bc)
              addArrayToEnv(c, lc)
         else
            addArrayToEnv(c, lc)
else
     opd := newopd•[ldEnv(c)]
    pc := pc +1
```

4 Conclusion

Array is an important feature in Java and JVM. The best and precise way to describe the essence of it is to build a model in mathematical way. In this paper, we define an extending model which uses an existing model as a basis and give the hierarchy of these two models, to model the array. In the extending model, we model the array in three steps.

Acknowledgment

The future work is to model the other important features in Java and JVM which have not been modeled by existing JVM . models.

References

1. Gosling, J., Joy, B., Steele, G.: The Java(tm) Language Specification. Addison-Wesley, Reading (1996)
2. Lindholm, T., Yellin, F.: The Java(tm) Virtual Machine Specification. Addison-Wesley, Reading (1996)
3. Borger, E., Schulte, W.: Modular Design for the Java Virtual Machine Archicture. In: Archicture Design and Validation Methods (2000)
4. Gurevich, Y.: Evolving algebras 1993: Lipari guide. In: Borger, E. (ed.) Specification and Validation Methods, Oxford University Press, Oxford (1995)
5. Borger, E., Schulte, W.: Defining the Java Virtual Machine as platform for provably correct Java compilations. In: Brim, L., Gruska, J., Zlatuška, J. (eds.) MFCS 1998. LNCS, vol. 1450, Springer, Heidelberg (1998)
6. Borger, E., Schulte, W.: A programmer friendly modular definition of the semantics of Java. In: Alves-Foss, J. (ed.) Formal Syntax and Semantics of Java(tm), Springer, Heidelberg (to appear, 1999)

Configuration Modeling Based Software Product Development

Yi-yuan Li, Jian-wei Yin*, Yin Li, and Jin-xiang Dong

College of Computer Science and Technology, Zhejiang Univ., Hangzhou 310027, China
zjulyy@yahoo.com.cn, zjuyjw@zju.edu.cn, cnliying@zju.edu.cn, djx@zju.edu.cn

Abstract. Software product line is an effective way to implement software production for mass customization. How to organize and configure the software artifacts in software product line to rapidly produce customized software product meeting individual demands is one of the key problems. Corresponding to the phases of feature selection and software artifact binding in the process of software production, the feature configuration model and software artifact configuration model are constructed to provide a uniform framework of constraint description for feature model and domain application requirement. The results of problem solving are the sets of feature and software artifact meeting feature constraints and application requirements. The proposed method of configuration modeling and problem solving provide a theoretical foundation to rapidly produce software product on the base of configuration of reusable domain assets.

Keywords: Feature configuration, Software artifact configuration, Configuration rule, Problem solving.

1 Introduction

Currently the manufacture of software engineering is suffering from such problems as individual customized requirements and frequent changes of business requirements. As a result, it seems that traditional software development mode - which is to develop software product specifically for certain application's requirements - costs more and has less efficiency and maintainability. In this software development mode, it's hard to meet the requirements of software development in large scale customization environment. The purpose of software production for mass customization is to produce and maintain a family of software products with similar functions, figure out both their commonalities and variability and manage these features [1]. It represents the trend of software factory's evolution.

Software product line is an effective way to implement software production for mass customization. It's a set of software systems with common controllable features [2]. The core idea of software product line engineering is to develop a reusable infrastructure that supports the software development of a family of products [3]. A

* Corresponding author.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 624– 639, 2007.

software product line typically consists of a product line architecture, a set of components and a set of products [4]. The characters of software development applying software product line principles are to maintain the common software assets and reuse them during the development process, such as domain model, software architecture, process model, components, and so on. Each product derives its architecture from the product line architecture, instantiates and configures a subset of the product line components and usually contains some product-specific code.

The purpose of developing software product family applying software product line principles is to rapidly produce individual customized software product with low cost and high quality. From the viewpoint of technique, how to organize and configure the assets of software product line to create a software product meeting the application requirements is one of the key problems to realize mass customization of software product. Product configuration is an important technique to solve this problem [5]. It can be represented as a design process to form an actual product based on a set of predefined components and the constraint relationships between them, which is generally an automatic or semi-automatic interactive decision-making process.

The abstract nature of the source codes makes the components which play as "parts" of the software product present the characteristics of non-standardization and variety during the development of software product. Classical product configuration model and problem solving methods is mostly oriented to such domain with uniform industry standard and interface definition as traditional manufacture [6, 7, 8]. It can't meet the demands of mass customization for software industry. Literatures [9, 10, 11] propose the translation of feature model into propositional formulas to be used for the automated analyses of feature model. D. Benavides et al use constraint programming to translate the feature model into a Constraint Satisfaction Problem(CSP) to get the potential software products [12, 13]. The above researches analyze the feature model by means of description of the feature constraints in logic. They ignore the effect on the software product development derived from the variability of software artifacts implementing the feature function. The configuration modeling and problem solving methods orienting software product line remain to be further researched.

This paper proposes to construct the models of feature configuration and software artifact configuration, which correspond to the phases of feature selection and software artifact binding in the process of software production respectively. The results of problem solving are the sets of feature and software artifact meeting application requirements. Thus the rapid production of software product on the base of configuration of reusable domain assets is available.

2 The Configuration Model for Software Product Line

In general, there are two relatively independent development cycles in organizations that apply software product line principles: product line development (domain engineering) and product instantiation (application engineering) [14]. On the base of domain analysis, domain engineering is responsible for the design, development and evolution of the reusable assets, such as architecture, reusable components and so on. Feature oriented domain analysis is the mainstream of the software product line modeling. Its main purpose is to identify all commonalities and variability in software product line. The outputs of modeling are all potential products of product line. Then

the domain reference architecture can be constructed from several logical layers, such as subsystem, process and module, to develop design model, process model and component entities realizing corresponding function of features. Application engineering, on the other hand, based on the requirement analysis, relates to adapt the product line architecture to fit the system architecture of the product in question, instantiate and configure a subset of the product line assets to reuse. It usually contains some product-specific code. Instantiated products constitute a family of software products in domain. The difference between software products in family represents the variability of software product line.

Thus it can be concluded the development of software product applying software product line principles is a process of feature implementation based on feature selection. It possesses the characteristic of dynamic. On one hand, because of the constraint on features, not all the features in feature model will be selected into the feature set oriented to specific application. On the other hand, there may exist more than one software artifact that implements the functions presented by a certain feature but with different quality of service for choices. Only those meeting specific non-functional requirements of application can be selected to construct the target system. The ultimate software product is the dynamic result of the selection of features and software artifacts to meet specific constraint and domain application demands.

2.1 Elementary Definition

Product configuration is a kind of design activity which takes the models of configuration and requirement as the input, takes the configuration result oriented to the ultimate product as the output. Hereinto configuration model describes the components and their relationships which may appear in the configuration result, requirement model describes the constraints that the ultimate product must meet, while the configuration result represents the selected components and their relationships in ultimate product. Then the process of software product development applying software product line principles can be regarded as a process of configuration modeling and problem solving which takes domain oriented feature model and reference architecture as configuration model, takes the domain application requirements oriented by software product as requirement model, takes the selection of features and its implementing artifacts under specific constraints as configuration rules, while takes customized software product oriented to concrete domain application as configuration result. This paper mainly researches the construction and problem solving of the software product line oriented configuration model.

2.2 Elementary Conception

Definition 1. The object according with the following definitions is called as ***term***:

(1) Constants are terms;

(2) Variables are terms;

(3) If $f(t_1,...,t_n)(n \geq 1)$ is a n-tuple function, in which $t_1,...,t_n$ are terms, then $f(t_1,...,t_n)$ is also a term;

(4) All the terms are created by using the above rules in finite times.

Definition 2. ***Atom*** can be represented as $p(a_1,...,a_n)$, where p is a n-tuple predication, and $a_1,...,a_n\ (n \geq 1)$ are terms. The atom having no variables is called as ***ground atom***, while the atom having variables is called as ***predication atom***.

Definition 3. The object according with the following definitions is called as ***literal***:

(1) Atoms are literals, an atom is satisfied if and only if its value equals true;

(2) Negative atoms are also literals. It is called as ***negative literal***. A negative literal is satisfied if and only if the value of the atom in it equals false.

Definition 4. ***Clause*** can be interpreted as the disjunction of several literals. It can be represented as follows:

$$not\ p_1 \vee ... \vee not\ p_m \vee q_1 \vee ... \vee q_n$$

Where both $not\ p_i$ and q_j are literals.

2.3 Configuration Rule

Rule is a representation of the configuration knowledge of software product line. It describes the constraint relationship that the principles of configuration knowledge must meet in the selecting process.

Definition 5. An ***elementary rule*** can be formatted as follows:

$$h \leftarrow p_1,...,p_m, not\ a_1,...,not\ a_n$$

Where $H(r)=\{h\}$ is called as rule head, $B^+(r)=\{p_1,...,p_m\}$ and $B^-(r)=\{not\ a_1,...,not\ a_n\}$ are called as positive literal and negative literal in rule body respectively, $B(r)=B^+(r)\cup B^-(r)$ is called as rule body. An elementary rule corresponds to a clause and can be interpreted as the disjunction of such literals as $not\ p_1,...,not\ p_m, a_1,...,a_n, h$, i.e. $not\ p_1 \vee ... \vee not\ p_m \vee a_1 \vee ... \vee a_n \vee h$. Thus an elementary rule is satisfied when the rule head h is satisfied or at least one literal in rule body is not satisfied.

Definition 6. A ***first order domain weight rule*** can be formatted as follows:

$$\forall \vec{x}\left(\begin{array}{l} LB_h \leq \left\{h_1\left(\vec{x_{h_1}}\right):Q_{h_1}=w_{h_1},...,h_s\left(\vec{x_{h_s}}\right):Q_{h_s}=w_{h_s}\right\} \leq UB_h \\ \quad \leftarrow LB_b \leq \left\{\begin{array}{l} p_1\left(\vec{x_{p_1}}\right):Q_{p_1}=w_{p_1},...,p_m\left(\vec{x_{p_m}}\right):Q_{p_m}=w_{p_m}, \\ not\ a_1\left(\vec{x_{a_1}}\right):Q_{a_1}=w_{a_1},...,not\ a_n\left(\vec{x_{a_n}}\right):Q_{a_n}=w_{a_n} \end{array}\right\} \leq UB_b \end{array}\right)$$

Where $\vec{x}$ is the variable vector in rule and can be omitted in ordinary condition; $h_i\left(\vec{x_{h_i}}\right)$, $p_j\left(\vec{x_{p_j}}\right)$, $a_k\left(\vec{x_{a_k}}\right)$ are predications, where $\vec{x_x}$ is the variable vector in predication and can be omitted when it is a constant; LB_x, UB_x are real number and

satisfy $LB_x \le UB_x$; w_{x_i} is also a real number. It is the weight value given to such predications as $h_i\left(\overrightarrow{x_{h_i}}\right)$, $p_j\left(\overrightarrow{x_{p_j}}\right)$ and $a_k\left(\overrightarrow{x_{a_k}}\right)$, and can be omitted when $w_{x_i}=1$; Q_{x_i} is a series of domain values used to restrict the range of such predications as $h_i\left(\overrightarrow{x_{h_i}}\right)$, $p_j\left(\overrightarrow{x_{p_j}}\right)$ and $a_k\left(\overrightarrow{x_{a_k}}\right)$. It can be signed as $Q_{x_i}=q_{x_i1}:q_{x_i2}:\ldots$. A first order domain weight rule is satisfied when $LB_h \le \sum_{h_i \text{ is satisfied, } 1\le i\le s} w_{h_i} \le UB_h$ or $LB_b \le \sum_{p_j \text{ is satisfied, } 1\le j\le m} w_{p_j} + \sum_{a_k \text{ is not satisfied, } 1\le k\le n} w_{a_k} \le UB_b$.

In the above definitions, the atoms in rules can correspond to the features in feature model, the variables can correspond to the software artifacts that implement the function of feature, domain is used to restrict the value range of variables in predications, w_{x_i} describes the value of the non-functional properties of the software artifacts bound to features, while LB_x and UB_x are used to limit the value range of the non-functional properties.

2.4 Configuration Model

Configuration constraints are the conditions the configuration result must satisfy. Its function is to make the configuration result valid. It can be described by the above rules. If regard the atoms and functions in rules as the principles of configuration knowledge and their function relationships, regard the rules as the constraint relationships used to select the principles of configuration knowledge, then the configuration knowledge of software product line can be described by logical program.

Definition 7. A logical program P is a set of elementary rules, each of which must be satisfied.

Definition 8. Program P represents the configuration model M of a software product. It is signed as $P(M)$. *Atoms* is the set of atoms in $P(M)$. If $C(M) \subseteq Atoms$ and $C(M) \models P(M)$, then $C(M)$ is called as a configuration of M, where $\models$ means logical restriction.

Definition 9. If P is a program, the mapping function $f_p : 2^{Atoms} \to 2^{Atoms}$ between two sets of atoms is defined as:

$$f_p(S) = \bigcup_{r\in P} f_r\left(S, f_p(S)\right)$$

Then $f_p : 2^{Atoms} \to 2^{Atoms}$ is a deduction closure on program P, where S is a subset of the atoms in P, f_r is a reasoning step on P for reduction. When all the atoms in

the rule are ground atoms, the first order domain weight rule can be interpreted by function $f_r : 2^{Atoms} \times 2^{Atoms} \to 2^{Atoms}$ as:

$$f_r(S,C) = \left(h \left| \begin{array}{l} LB_h \le \left| \sum_{h_i \in Atoms(H(r)) \cap S} w_{h_i} \right| \le UB_h, \\ LB_b \le \left| \sum_{p_j \in Atoms(B^+(r)) \cap C} w_{p_j} + \sum_{a_k \notin Atoms(B^-(r)) \cap S} w_{a_k} \right| \le UB_b \end{array} \right. \right)$$

Where C is a deduction closure on program P_S, which is a program reasoned from P after reduction. The function $g_p(S)$ is defined as:

$$g_p(S) = \cap \left\{ f_p(S) \middle| f_p : 2^{Atoms} \to 2^{Atoms} \text{ is a deduction closure on } P \right\}$$

Then if and only if $S = g_p(S)$, S is a stable model of P.

The visual meaning of the definition is that when use the subset S of the atoms in P to reduce each rule in program, the program P is simplified to P_S. If the deduction closure on P_S accords with S, then S is a configuration of the configuration model M represented by P.

3 Configuration Problem Solving

The development process of software product applying software product line principles can be divided to the phase of feature selection under the feature constraint and the phase of software artifact binding. The former constructs the feature set and their relationships meeting the constraints in terms of the functional requirements of application based on the feature model, while the latter gets the software artifacts implementing the selected features to assembly a software product. Corresponding to the two phases of software development, the software product line oriented product configuration can be separated to the phase of feature configuration and the phase software artifact configuration, which obtain the set of features and the set of software artifacts meeting the application demands respectively.

3.1 Problem Solving of Feature Configuration

The problem solving of feature configuration corresponds to the process of feature selection, where features and their constraint relationships are the primary objects being considered. In this condition, the instantiation of the variables in the rules is not taken into account for the moment, that is to say, all the atoms in the rules are ground atoms. The problem solving process of feature configuration mainly includes the following steps:

(1) Construct the set of configuration rules in terms of the application demands and feature model;

(2) Solve the configuration rules to get the feature set meeting the functional requirements of specific domain application.

3.1.1 Construct the Rules of Feature Configuration

If not take the instantiation of the variables in the rules into account temporarily, some elementary rules of feature configuration are formed by setting values to such parameters as w_{x_i}, LB_x and UB_x in the first order domain weight rule.

Definition 10. An ***inconsistency rule*** (*IR*) can be formatted as follows:

$$\leftarrow p_1,...,p_m,not\ a_1,...,not\ a_n$$

It can be derived from the first order domain weight rule in the condition $w_{x_i}=1$, $LB_h=UB_h=0$, $LB_b=1$, $UB_b=m+n$. That is to say, the rule head of the inconsistency rule is empty. When at least one literal in the rule body is not satisfied, the inconsistency rule is satisfied. In the process of feature configuration, the visual meaning of the inconsistency rule is that the condition that the feature set $\{p_1,...,p_m\}$ appears in the configuration result, while the feature set $\{a_1,...,a_n\}$ does not appear in the configuration result at the same time impossibly takes place.

Definition 11. A ***reality rule*** (*RR*) can be formatted as follows:

$$h\leftarrow$$

It can be derived from the first order domain weight rule in the condition $w_{x_i}=1$, $LB_h=UB_h=1$, $LB_b=UB_b=0$. That is to say, the rule body of the reality rule is empty. When the rule head h is satisfied, the reality rule is satisfied. In the process of feature configuration, the visual meaning of the reality rule is that the feature h must appear in the configuration result.

Definition 12. A ***choice rule*** (*CR*) can be formatted as follows:

$$h_1\,|\,...\,|\,h_s\leftarrow p_1,...,p_m,not\ a_1,...,not\ a_n$$

It can be derived from the first order domain weight rule in the condition $w_{x_i}=1$, $LB_h=1$, $UB_h=s$, $LB_b=1$, $UB_b=m+n$. When at least one atom in $\{h_1,...,h_s\}$ is satisfied or at least one literal in the rule body is not satisfied, the choice rule is satisfied. In the process of feature configuration, the visual meaning of the choice rule is that if the feature set $\{p_1,...,p_m\}$ appears in the configuration result and the feature set $\{a_1,...,a_n\}$ does not appear in the configuration result, then a subset of $\{h_1,...,h_s\}$ must be selected to appear in the configuration result. The reduction result of choice rule can be represented as an elementary rule with the format $h_i\leftarrow p_1,...,p_m,not\ a_1,...,not\ a_n$, where $h_i\in\{h_1,...,h_s\}\cap S$.

Definition 13. An ***exclude rule*** (ER) can be formatted as follows:

$$h_1 \oplus ... \oplus h_s \leftarrow p_1, ..., p_m, not\ a_1, ..., not\ a_n$$

It can be derived from the first order domain weight rule in the condition $w_{x_i} = 1$, $LB_h = UB_h = 1$, $LB_b = 1$, $UB_b = m + n$. When only one atom in $\{h_1, ..., h_s\}$ is satisfied or at least one literal in the rule body is not satisfied, the exclude rule is satisfied. In the process of feature configuration, the visual meaning of the exclude rule is that if the feature set $\{p_1, ..., p_m\}$ appears in the configuration result and the feature set $\{a_1, ..., a_n\}$ does not appear in the configuration result, then a feature in $\{h_1, ..., h_s\}$ must be selected to appear in the configuration result. The reduction result of exclude rule can be represented as an elementary rule with the format $h_i \leftarrow p_1, ..., p_m, not\ a_1, ..., not\ a_n$, where $\left|\{h_1, ..., h_s\} \cap S\right| = 1$.

The rules of feature configuration consist of two parts, one is the feature model oriented configuration rule, and the other is the domain application requirement oriented configuration rule.

The feature model oriented configuration rule is used to describe the feature constraints and can be represented with the above elementary rules of feature configuration and their combination. For instance, the exclusive relationship of two child features with different parent feature can be represented by the inconsistency rule, such as the condition that features A and B are exclusive and the parent feature of A is not that of B can be described as $\leftarrow A, B$. The exclusive relationship of two child features with same parent feature can be represented by the exclude rule, such as the condition that features A and B are exclusive and both of them are the optional child features of feature C can be described as $A \oplus B \leftarrow C$. It is reduced to $B \leftarrow C$ if the child feature B is selected. The mandatory feature can be represented by the elementary rule or the reality rule, such as the condition that the child feature B of the feature A is a mandatory feature can be described as $B \leftarrow A$. The optional feature can be represented by the choice rule, such as the condition that the features B, C, D are the optional child features of feature A can be described as $B \mid C \mid D \mid \varnothing \leftarrow A$. It is reduced to $B \leftarrow A$ if the child feature B is selected.

The domain application requirement oriented configuration rule is used to describe the functional requirement of specific application and can be represented with a set of reality rules. For instance, if function f is asked to realize by some domain application, and it is modeled as an optional feature F in the feature model during the process of domain analysis, then this requirement can be described as $F \leftarrow$. The feature model oriented configuration rule and the domain application requirement oriented configuration rule provide the precondition and the target for the problem solving of feature configuration respectively.

3.1.2 Problem Solving of the Rules of Feature Configuration

Let P denotes the logical program corresponding to feature model, R denotes the rule set of feature configuration in P , A denotes a expanding set with the initiated

value $\varnothing$ and satisfies $A^+ \cup A^- = Atoms(A)$, where $A^+ = \{a \in Atoms(P) | a \in A\}$, $A^- = \{a \in Atoms(P) | not\ a \in A\}$, then the steps of problem solving are described as follows:

Step1: Get all the reality rules in the rule set R and put the literals in rule head into the expanding set A, i.e. $\forall r \in RR$, let $A = A \cup H(r)$, $R' = R$;

Step2: Let $A' = A$, and for each non reality rule r in the rule set R', designate r' is the reduced result of r after reasoning, then

(1) if there exists no negative literals in the rule body and the atoms of the positive literals appear in A^+, then put the literals in the rule head of the reduced rule into the expanding set A and replace the original rule with the reduced rule, i.e. if $B^-(r) = \varnothing$, and $\forall a \in Atoms(B^+(r))$ satisfies $a \in A^+$, then let $A = A \cup H(r')$, $R' = R' - \{r\} + \{r'\}$;

(2) if there exists negative literals in the rule body and the atoms of the negative literals appear in A^+, then delete this rule from the rule set R', i.e. if $\exists a \in Atoms(B^-(r))$ satisfies $a \in A^+$, then let $R' = R' - \{r\}$;

(3) if the atoms of the positive literals in the rule body appear in A^+, and the atoms of the negative literals in the rule body do not appear in A^+, then no matter whether the atoms of the negative literals in rule body appear in A^-, put the literals in the rule head of the reduced rule into the expanding set A and replace the original rule with the reduced rule, i.e. if $\forall a \in Atoms(B^+(r))$ satisfies $a \in A^+$ and $\forall a \in Atoms(B^-(r))$ satisfies $a \notin A^+$, then let $A = A \cup H(r')$, $R' = R' - \{r\} + \{r'\}$;

(4) if the rule head is satisfied, for instance, toward exclude rule, $\exists a \in H(r)$ satisfies $a \in A$, toward choice rule, $\exists H' \subseteq H(r)$ satisfies $H' \subseteq A$ and $|Atoms(H')| \geq 1$, then put the literals in rule body into the expanding set A and replace the original rule with the reduced rule, i.e. $\forall a \in Atoms(B^+(r))$, let $A = A \cup \{a\}$, and $\forall a \in Atoms(B^-(r))$, let $A = A \cup \{not\ a\}$, $R' = R' - \{r\} + \{r'\}$;

Step3: repeat *Step2* until $A = A'$;

Step4: if $A^+ \cap A^- \neq \varnothing$, then it can be concluded that there is no stable solving model for the feature configuration model and the solving process finishes. Contrarily if $Atoms(R') \subseteq Atoms(A)$, then A^+ is the stable solving model.

Otherwise let $R' = R$, and $A = A \cup \{a\}$ or $A = A \cup \{not\ a\}$, where $a \in Atoms(R) - Atoms(A)$, then re-execute *Step2*.

The visual meaning of the solving arithmetic is that the literals in the rule head of the reality rules are regarded as facts to be put into the configuration result, while the rest rules are reasoned to expand the result set on the condition that the rule body or the rule head is satisfied. During the process, conflicts need to be detected. The condition that the conflict exists denotes that there are no stable solving model that accords with A. If there is no conflict and A covers the atom set of the simplified logical program after the reduction of the rules, the atom set of the positive literals in A is the configuration result. Otherwise A need to be tentatively expanded to be further validated.

3.2 Problem Solving of Software Artifact Configuration

The problem solving process of software artifact configuration mainly includes the following steps:

(1) Construct the set of configuration rules in terms of the feature set resulting from the problem solving of feature configuration;

(2) Solve the configuration rules to get the software artifact set meeting the requirements of specific domain application.

3.2.1 Construct the Rules of Software Artifact Configuration

The rules of software artifact configuration are constructed on the base of the problem solving of the feature configuration. It is a process of getting software artifacts to set up variables of feature implementation in terms of the result set of feature configuration and confirming the value range of variables and the bound of weight constraint in terms of the non-functional requirements of specific domain application.

The atoms in the rule set of the logical program P corresponding to the feature model represent the features. There may exist several optional schemes and software artifacts to implement the function presented by the feature, so the atoms in the rules of software artifact configuration are not the ground atoms p but the predication atoms $p(\vec{x})$ having the variables corresponding to the software artifacts. The meaning of the predication $p(\vec{x})$ can be described as the function represented by feature p is implemented by the software artifact vector $\vec{x}$. The software artifacts implementing the same feature function may have distinct quality of service, such as performance, resource consuming, and so on. The selection of the software artifacts rests with the non-functional requirements of the specific domain application.

Thus the construction process of the rules of software artifact configuration is composed of several steps as follows, including reducing the rule set of feature configuration, confirming the value range of variables, instantiating the variables and confirming the bound of weight constraint:

Step1: For each rule in the reduced rule set R' derived from the problem solving of feature configuration, if there exists negative literal in the rule body and the atoms

in the negative literal appear in A^+, then delete the rule, i.e. $\forall r \in R'$, if $B^-(r) \neq \varnothing$ and $a \in Atoms(B^-(r))$ satisfied $a \in A^+$, then $R' = R' - \{r\}$, and delete the negative literals in the rule body of the rest rules, i.e. $\forall r \in R'$, let $B^-(r) = \varnothing$. Thus there are no negative literals in the rule set of software artifact configuration;

Step2: Get the corresponding software artifacts for each predication atom, namely feature, in the configuration result A^+ of feature configuration;

Step3: Confirm the constants or the variables in the rule set. If the feature represented by the predication atom has multiple software artifacts to implement the function, then it can be denoted by a predication atom with variable, otherwise, it is denoted by a ground atom, the constant in which represents the software artifact implementing the feature function;

Step4: For each predication atom with variable, get the candidate software artifacts implementing the corresponding feature function to determine the value range of the variable;

Step5: Confirm the weight value of the corresponding predication atom in terms of the non-functional attribute value of the software artifact;

Step6: Confirm the range of weight constraint in terms of the non-functional requirements of specific domain application to build the choice constraint of the software artifacts under the general restriction.

3.2.2 Problem Solving of the Rules of Software Artifact Configuration

Definition 14. Each constant or the function of the constants in logical program P is called a ground term of P. $GTerms$ denotes the set of the ground terms. If all the variables in the logical program P are replaced by the ground terms, then the result program after replacement is called the instantiation of P.

Let P denotes the logical program corresponding to configuration model of software artifact, R denotes the rule set of software artifact configuration in P, T is the configuration result of software artifacts, F is the feature set implemented by the software artifacts. The values of T and F are initiated to $\varnothing$, then the steps of problem solving are described as follows:

Step1: Get the set of all the variables and the predication atoms the variables pertaining to in the rule set, which are signed as V and D respectively;

Step2: Let $T' = T$, $F' = F$, for each rule with no variables and weight values in the rule set,

(1) if the rule is a reality rule and the ground atoms in the rule head do not appear in D, then put the ground atoms and their ground terms in the rule head into the set F and T respectively, and delete the rule from the rule set. i.e. $\forall r \in NR$, if $\forall a \in Atoms(H(r))$ satisfies $a \notin D$ and $w_a = 1$, then let $F = F \cup H(r)$, $T = T \cup GTerms(H(r))$, $R = R - \{r\}$;

(2) If the ground atoms in the rule do not appear in D and the ground atoms and their ground terms in rule body appear in the set F and T respectively, then put the ground atoms and their ground terms in the rule head into the set F and T

respectively, and delete the rule from the rule set. i.e. $\forall r \notin NR$, if $\forall a \in Atoms(r)$ satisfies $a \notin D$ and $w_a = 1$, $\forall a \in Atoms(B(r))$ and $\forall t \in GTerms(B(r))$ satisfies $a \in F$ and $t \in T$, then let $F = F \cup H(r)$, $T = T \cup GTerms(H(r))$, $R = R - \{r\}$;

Step3: Instantiate the variables in the rule set in terms of the value range and delete the instantiated variables and the predication atoms pertained to by the variables from the set V and D respectively. Then replace the original rule with the instantiated rules. Each variable has a series of domain values, so the rule with variables can be instantiated as more than one ground term. As a result, several rule sets are achieved after the instantiation of all the variables. Let n represents the amount of the rules with predication atom in the rule set R, then $\forall r_i \in R$, $1 \le i \le n$, if $Inst_i$ represents the amount of the achieved rules after the rule r_i is instantiated, then the amount of the achieved rule sets after the rule set R is instantiated is $\prod_{i=1}^{n} Inst_i$. All the atoms in the instantiated rule sets are ground atoms;

Step4: In each rule set after instantiation, for each rule with the weight constraint,

(1) if the ground atoms and their ground terms in the rule body appear in the set F and T respectively, and the weight value meets the corresponding weight constraint, then set the rule body as empty to be a reality rule and replace the original rule in the rule set, i.e. if $\forall a \in Atoms(B(r))$ satisfies $a \in F$, $\forall t \in GTerms(B(r))$ satisfies $t \in T$, and $LB_b \le \sum_{p_j \text{ is satisfied, } 1 \le j \le m} w_{p_j} \le UB_b$, then let $r' = r$, $B(r') = \varnothing$, $R = R - \{r\} + \{r'\}$, otherwise, delete the ground atoms and their ground terms in the rule body from the set F and T respectively, i.e. if $\sum_{p_j \text{ is satisfied, } 1 \le j \le m} w_{p_j} < LB_b$ or $\sum_{p_j \text{ is satisfied, } 1 \le j \le m} w_{p_j} > UB_b$, let $F = F - Atoms(B(r))$, $T = T - GTerms(B(r))$;

(2) if the rule body is empty, the ground atoms and their ground terms in the rule head appear in the set F and T respectively, and the weight value meet corresponding weight constraint, then do nothing, otherwise, delete the ground atoms and their ground terms in the rule head from the set F and T respectively, i.e. if $\sum_{h_i \text{ is satisfied, } 1 \le i \le s} w_{h_i} < LB_h$ or $\sum_{h_i \text{ is satisfied, } 1 \le i \le s} w_{h_i} > UB_h$, let $F = F - Atoms(H(r))$, $T = T - GTerms(H(r))$;

Step5: Repeat *Step2* and *Step4* until $F = F'$ and $T = T'$;

Step6: Let A^+ denotes the result set of problem solving of feature configuration in chapter 2.1.2, if $F = A^+$, then T is the result set of problem solving of software artifact configuration, otherwise, it represents that there exists no valid result set of

software artifacts meeting the non-functional requirements of specific domain application.

The visual meaning of the solving arithmetic is that if the scheme of feature implementation is unique, then the corresponding software artifacts are regarded as the realities to be added into the configuration result, otherwise, the weight constraint of the rest rules are examined to be met or not on the base of the instantiation of the variables representing the software artifacts in terms of the candidate schemes of feature implementation. If the obtained feature set corresponded by the set of software artifacts accords with the result set of feature configuration, then it denotes that there exists valid result set of software artifacts meeting the non-functional requirements, otherwise, it shows the existing scheme of feature implementation can't meet the non-functional requirements of domain application.

4 Case Study

Figure 1 shows an expanded feature model of mobile phone software product line. It is composed of some functional features like *game*, *password protection*, *contact list* and *web browser* etc. Among them, *password protection* and *web browser* is optional features. Meanwhile, multiple games can be the choice, but to some limitation, such as a small memory capacity, *G3* and *G4* can only be chosen one arbitrarily. In the process of feature analysis, each function feature has related requirements specification, design model and implementation component. Some functional features, for example, *G2*, even have various implementation schemes with distinct quality of service.

If *web browser* and *G3* are required by some specific application, then in terms of the feature model of mobile phone software product line and the functional demands, the rule set R of the feature configuration is listed as follows:

$$MP \leftarrow, \quad WB \leftarrow, \quad G3 \leftarrow, \quad GS \leftarrow MP,$$
$$CL \leftarrow MP, \quad WB \mid PWP \mid \varnothing \leftarrow MP, \quad G1 \leftarrow GS, \quad G2 \leftarrow GS,$$
$$G3 \oplus G4 \leftarrow GS, \quad G4 \leftarrow not\ \mathrm{G3}, \quad G3 \leftarrow not\ \mathrm{G4}$$

According to the solving steps of feature configuration, $A = \{MP\}$ is available after *Step1* is executed. Similarly there exists $R' = \left\{ \begin{matrix} MP \leftarrow, WB \leftarrow, G3 \leftarrow, GS \leftarrow MP, CL \leftarrow MP, \\ WB \leftarrow MP, G1 \leftarrow GS, G2 \leftarrow GS, G3 \leftarrow GS, G3 \leftarrow not\ \mathrm{G4} \end{matrix} \right\}$ and $A = \{MP, GS, CL, WB, G1, G2, G3, not\ G4\}$ after executing *Step2* and *Step3*. Then the condition that $Atoms(R') \subseteq Atoms(A)$ and $A^+ \cap A^- \neq \varnothing$ are satisfied, so the set $A^+ = \{MP, GS, CL, WB, G1, G2, G3\}$ is a stable model for the rule set of feature configuration.

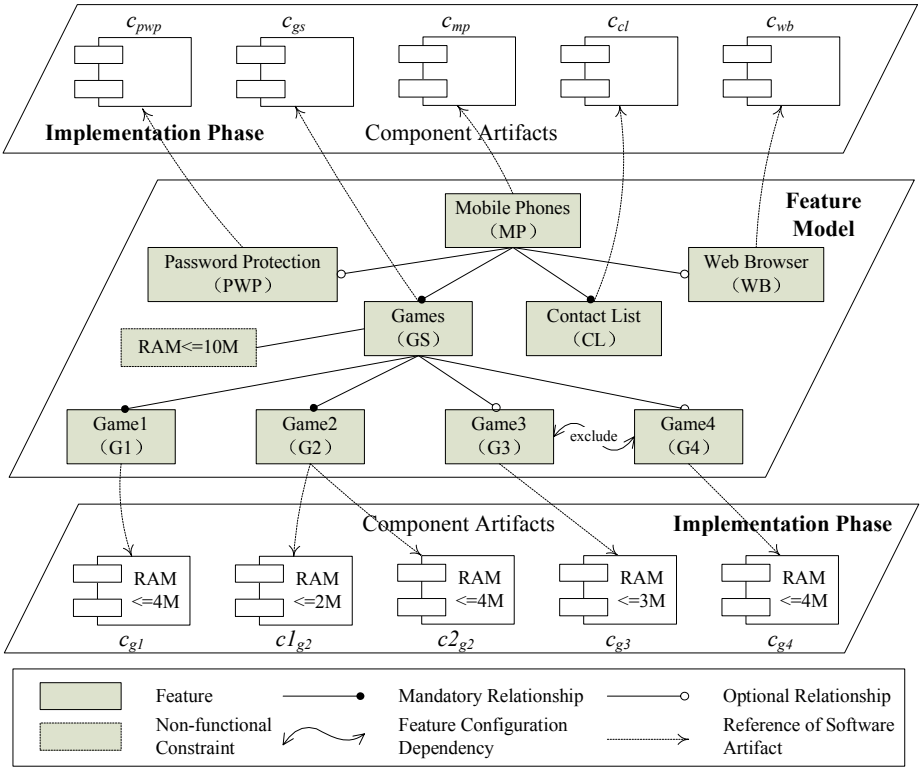


Fig. 1. Expanded Feature Model of Mobile Phone Software Product Line

If the total memory space consumed by the games is asked to no more than *10M* to make the software system run normally, then the rule set of software artifact configuration is constructed on the base of the result set of problem solving of feature configuration and listed as below in the condition that only component artifacts are taken into account:

$$MP\left(c_{mp}\right) \leftarrow , \qquad WB\left(c_{wb}\right) \leftarrow MP\left(c_{mp}\right), \quad CL\left(c_{cl}\right) \leftarrow MP\left(c_{mp}\right),$$

$$GS\left(c_{gs}\right) \leftarrow MP\left(c_{mp}\right), \qquad G1\left(c_{g1}\right) \leftarrow GS\left(c_{gs}\right),$$

$$G2\left(\vec{x}\right) : c1_{g2} : c2_{g2} \leftarrow GS\left(c_{gs}\right), \qquad G3\left(c_{g3}\right) \leftarrow GS\left(c_{gs}\right),$$

$$0 \le \left\{G1\left(c_{g1}\right) = 4, G2\left(\vec{x}\right) : c1_{g2} = 2 : c2_{g2} = 4, G3\left(c_{g3}\right) = 3\right\} \le 10 \leftarrow GS\left(c_{gs}\right)$$

According to the solving steps of software artifact configuration, $V = \left\{\vec{x}\right\}$ and $D = \left\{G2\right\}$ are set up after *Step1* is executed. It satisfies $F = \left\{MP, WB, CL, GS, G1, G3\right\}$ and $T = \left\{c_{mp}, c_{wb}, c_{cl}, c_{gs}, c_{g1}, c_{g3}\right\}$ after executing *Step2*. If *Step3* is executed, $V = \varnothing$ and $D = \varnothing$ are established, and

the rule $G2\left(\bar{x}\right):c1_{g2}:c2_{g2}\leftarrow GS\left(c_{gs}\right)$ is instantiated as $G2\left(c1_{g2}\right)\leftarrow GS\left(c_{gs}\right)$ and $G2\left(c2_{g2}\right)\leftarrow GS\left(c_{gs}\right)$, the rule $0\le\left\{G1\left(c_{g1}\right)=4,G2\left(\bar{x}\right):c1_{g2}=2:c2_{g2}=4,G3\left(c_{g3}\right)=3\right\}\le 10\leftarrow GS\left(c_{gs}\right)$ is instantiated as $0\le\left\{G1\left(c_{g1}\right)=4,G2\left(c1_{g2}\right)=2,G3\left(c_{g3}\right)=3\right\}\le 10\leftarrow GS\left(c_{gs}\right)$ and $0\le\left\{G1\left(c_{g1}\right)=4,G2\left(c2_{g2}\right)=4,G3\left(c_{g3}\right)=3\right\}\le 10\leftarrow GS\left(c_{gs}\right)$. The instantiated rules are combined to four rule sets, two of which have no resolution, while the other two can be deduced to get the conclusion that $F_{c1_{g2}}=\left\{MP,WB,CL,GS,G1,G3,G2\right\}$, $T_{c1_{g2}}=\left\{c_{mp},c_{wb},c_{cl},c_{gs},c_{g1},c_{g3},c1_{g2}\right\}$ or $F_{c2_{g2}}=\left\{MP,WB,CL,GS\right\}$, $T_{c2_{g2}}=\left\{c_{mp},c_{wb},c_{cl},c_{gs}\right\}$ respectively after *Step4* and *Step5* are executed. Obviously $F_{c1_{g2}}=A^{+}$ is satisfied, so $T_{c1_{g2}}$ is the result set of software artifact configuration meeting the requirements of specific domain application.

5 Conclusion

How to organize and configure the assets in software product line to rapidly produce a software product meeting the individual requirement is a key problem to realize the mass customization of software product applying software product line principles. Based on the analysis of the development process of software product applying software product line principles, this paper propose the concept of software product line oriented product configuration. Corresponding to the phases of feature selection and software artifact binding in the process of software production, the feature configuration model and software artifact configuration model are constructed to provide a uniform framework of constraint description for feature model and domain application requirement. The results of problem solving are the sets of feature and software artifact meeting feature constraints and application requirements. The division of the product configuration to the models of feature configuration and software artifact configuration accords with the development process of software product applying software product line principles in logic. It can reduce the size of the rule set and decrease the complexity of the problem solving to increase efficiency. The case study of the mobile phone software product line illustrates the construction and problem solving process of the software product line oriented configuration model. The further work is to refine the arithmetic of problem solving to reuse the configuration result and investigate the construction and problem solving of software product line oriented product configuration in distributed collaborative environment.

Acknowledgement

The work has been supported by the National High-Tech. R&D Program, China (No.2006AA01Z170, No.2006AA01Z171, No.2007AA01Z124) and the highlight R&D Program of Zhejiang Province (No. 2006C11206).

References

1. Krueger, C.W.: Software Mass Customization. BigLever Software, Inc. (2001)
2. Clements, P.C., Northrop, L.: Software Product Lines - Practices and Patterns. Addison-Wesley, Reading (2001)
3. Jaring, M., Bosch, J.: Representing Variability in Software Product Lines: A Case Study. In: Chastek, G.J. (ed.) Software Product Lines. LNCS, vol. 2379, pp. 15–36. Springer, Heidelberg (2002)
4. Bosch, J.: Design & Use of Software Architectures - Adopting and Evolving a Product-Line Approach. Addison-Wesley, Reading (2000)
5. Bourke, R.: Product Configurators: Key Enabler for Mass Customization - An Overview (2000), http://www.pdmic.com/articles/midrange/Aug2000.html
6. Samson, W., Henrik, E., Gennari, J.: Ontology-Based Configuration of Problem-Solving Methods and Generation of Know PROTÉGÉ-II to Protocol-Based Decision Support". Artificial Intelligence in Medicine 7, 257–289 (1995)
7. Soininen, T., Tiihonen, J., Mannisto, T.: Towards a General Ontology of Configuration. AI/EDMS 12(4), 357–372 (1998)
8. Studer, R., Eriksson, H., Gennari, J.H.: Ontologies and The Configuration of Problem-Solving Methods. In: Proceedings of 10th Knowledge Acquisition for Knowledge-base Systems Workshop, Banff (1996)
9. Mannion, M.: Using First-Order Logic for Product Line Model Validation. In: Chastek, G.J. (ed.) Software Product Lines. LNCS, vol. 2379, pp. 176–187. Springer, Heidelberg (2002)
10. Sun, J., Zhang, H., Li, Y.F., Wang, H.: Formal Semantics and Verification for Feature Modeling. In: ICECSS 2005 (2005)
11. Zhang, W., Zhao, H., Mei, H., Propositional, A.: Logic-based Method for Verification of Feature Models. In: Davies, J., Schulte, W., Barnett, M. (eds.) ICFEM 2004. LNCS, vol. 3308, pp. 115–130. Springer, Heidelberg (2004)
12. Benavides, D., Ruiz-Cortés, A., Smith, B., O'Sullivan, B., Trinidad, P.: Computational Issues on the Automated Analysis of Feature Models Using Constraint Programming. International Journal of Software Engineering and Knowledge Engineering (2006)
13. Benavides, D., Ruiz-Cortés, A., Trinidad, P.: Using Constraint Programming to Reason on Feature Models. In: Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering (2005)
14. Kang, K.C., Kim, S., Lee, J., Kim, K., Kim, G.J., Shin, E.: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering 5, 143–168 (1998)

Formal Semantic Meanings of Architecture-Centric Model Mapping

Xiao Yang, Jinkui Hou, and Jiancheng Wan

School of Computer Science and Technology, Shandong University,
Jinan, 250061, China
{yangx,houjk}@mail.sdu.edu.cn, wanjch@sdu.edu.cn

Abstract. Over the past few years, Model-driven Development (MDD) has become an active research area of software engineering, in which model transformation is a key technology. However, there is currently no mature foundation on the definition of mapping rules as well as cardinal principles to verify the mapping relations between such models. Based on software architecture, category theory is used to explore the mapping relations between models at different abstract levels, so that the interconnections and mapping relations between component-based models and the compositions of these relations have rigorous meanings. The morphism composition and functors are used to trace the relationships between component models at different abstract levels. Formal description of model mappings is suitable to the automatic software development. It can be a measurement for validating the mapping rules between different models, and thus can make an effective support to MDD.

1 Introduction

Over the past few years, Model-driven Development (MDD) has become an active research area of software engineering [1], in which model transformation is a key technology. Represented by OMG's MDA, numerous research institutions and enterprises have been investing a large amount of money and manpower in the model transformation study. Currently, a number of products based on MDA have proved that a lot of benefits can be obtained from it, such as rapid development, architecture advantages, improvement of code consistency and maintainability, enhancement of system's portability across middleware vendors, and it also shows great potential in these areas.

Most of the existing and proposed approaches [2] for model transformation focus on providing a concrete solution for the transformation between models at different abstract levels, and there's currently no mature foundation on the definition of mapping rules as well as cardinal principles to validate the mapping relations between such models. More in-depth study and formal methods about this issue are expected to support complicated model transformation [3]. Consequently, a unifying framework for the mapping description techniques seems imperative. Such a framework should be formal, in order to avoid ambiguities; offer a sufficiently high

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 640–649, 2007.

level of abstraction, in order to concentrate on the meaning of concepts instead of on representational aspects; and be sufficiently expressive. These requirements suggest category theory as an excellent candidate.

Category theory is a mathematical framework suitable for representing relationships between knowledge [4], which has been viewed by the computer sciences as a means of achieving representation independence and abstraction, while providing conceptual subdiscipline unification [5]. Category theory has been widely used to facilitate specification construction [6]. It provides the right level of mathematical abstraction to address languages for describing software architectures [7]. The abstract framework of category theory is shown to provide semantics for the configuration of complex systems from their component parts. Diagrams express configuration by representing the results of applying combinators to recursively defined system components. These ideas are extended to provide a precise semantics for both components structuring and models mapping in this paper. We propose adaptations to the categorical framework in order to manage model mapping and transformation.

The rest of this paper is organized as follows: some basic concepts of category and algebraic specification are given briefly in Section 2; the formal semantic meanings of component model mapping are developed in Section 3; a case study about email client model mapping is shown in Section 4; related works in this area are presented in Section 5; The paper ends with conclusions and future works.

2 Category Theory and Algebraic Specification

In computing science, more abstract viewpoints are often more useful, because of the need to achieve independence from the overwhelmingly complex details of how things are represented or implemented. Category theory allows the study of the essence of certain concepts as it focuses on the properties of mathematical structures instead of on their representation. One of the basic principles summarized in [8] is that complex systems can be usefully identified with diagrams, system components and connectors corresponding to nodes, and interconnections being established through the edges of the diagrams. Category theory is ideal for this purpose, as it provides a rich body of theory for reasoning about objects and relations between them. Moreover, category theory lends itself well to automation, so that, for example, the composition of two specifications can be derived automatically, provided that the category of specifications obeys certain properties. Most of the category definitions of this section are adapted from [4].

Definition 1. Category. A category $\mathbb{C}$ is composed of two collections:

(1) the objects of the category, which is called *C*-objects;

(2) the morphisms (arrows) of the category, which is called *C*-arrows;

These two collections must respect the following properties:

(a) each morphism f is associated with an object *A* that is its domain and an object B that is its codomain. Notation: $f: A \rightarrow B$.

(b) for all morphisms $f: A \rightarrow B$ and $g: B \rightarrow C$, there exists a composed morphism $g \circ f: A \rightarrow C$ and the composition law is associative, i.e. for all $h: C \rightarrow D$, $h \circ (g \circ f) = (h \circ g) \circ f$.

(c) for each object A of the category, there exists an identity morphism id_A such that:

$$\forall f\colon B \to A,\ id_A \circ f = f \text{ and } \forall f\colon A \to B;\ f \circ id_A = f.$$

Many categorical definitions and proofs employ diagrams. As remarked before, quite complex facts can be visualized by the use of these diagrams.

Definition 2. Diagram. A diagram $\mathbb{D}$ in a category $\mathbb{C}$ consists of a collection D_C of C-objects and a collection D_A of C-arrows such that for any arrow $a \in D_A$, $cod\ a \in D_C$ and $dom\ a \in D_C$, where *cod a* represents the codomain of a and *dom a* represents the domain of a.

Definition 3. Commutative diagram. A diagram is said to *commute* if every path between two objects in its image determines through composition the same arrow. The case is shown in Fig.1.

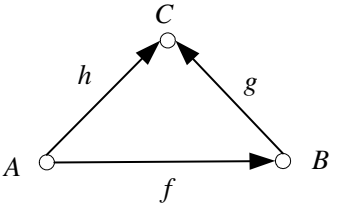


Fig. 1. Diagram commutes iff h is the composite g ∘ f

A powerful construction operation called colimit is defined over diagrams.

Definition 4. Colimit. A colimit for a diagram $\mathbb{D}$ in a category $\mathbb{C}$ is a C-object C along with a co-cone$\{f_i : D_i \to C \mid D_i \in \mathbb{D}\}$ from **D** to C such that for any other co-cone $\{f'_i\colon D_i \to c' \mid D_i \in \mathbb{D}\}$ from **D** to a vertex C', there is a unique C-arrow $f : C \to C'$ such that for every object D_i in $\mathbb{D}$, the diagram shown in Fig.2. commutes; i.e., $f \circ f_i = f'_i$.

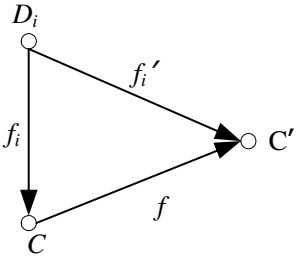


Fig. 2. Definition of a colimit

A practical interpretation for the colimit is given by Goguen in [5]: "Given a species of structure, say widgets, then the result of interconnecting a system of widgets to form a super-widget corresponds to taking the colimit of the diagram of widgets in which the morphisms show how they are interconnected."

Definition 5. Functor. A functor F from a category $\mathbb{C}$ to a category $\mathbb{D}$ is a function which assigns to each C-object a, a D-object $F(a)$, and to each C-arrow f: $A \rightarrow B$, a D-arrow $F(f)$: $F(A) \rightarrow F(B)$, such that identity arrows and composites are preserved, i.e., $F(id_A) = id_{F(A)}$; for all C-objects A, and $F(g \circ f) = F(g) \circ F(f)$; whenever $g \circ f$ is defined in $\mathbb{C}$.

Category theory can be used to compose formal specifications from smaller, reusable pieces. When used for specification construction, there is usually a requirement that the morphisms preserve theoremhood. That is, if a morphism between two specifications is defined, there is an obligation to prove that the axioms of the source specification are theorems of the target specification under the translation. Thus we can define an architecture model as a diagram of specifications, and prove properties of this architecture at a relatively abstract level.

3 Formal Semantic Meanings for Architecture Model

In MDD, the model description must be precise enough to grasp the essential behavior of the component, which also must be sufficiently abstract to ensure that, according to the requirements of the model, different vendors can respectively develop their component products that can compete with each other. A category theoretic foundation is shown in this section for the conceptual component modeling elements.

3.1 Component Signature and Component Specification

From a mathematical point of view, component signatures are structures defined as follows.

Definition 6. Component signature. A component signature is a 6-tuple $<\Sigma, A, \Gamma, fa, fp, D>$ where

(1) $\Sigma = <S, \Omega>$ is a data signature in the usual algebraic sense, i.e. a set S of sort symbols and a $S^* \times S$-indexed family Ω of function symbols;

(2) A is a $S^* \times S$-indexed family of attribute symbols of the component, each attribute is typed by a data sort in S;

(3) Γ is an S^*-indexed family of port symbols.

(4) fa: $A \rightarrow S_A$, $S_A \subset S$ is a total function, which shows the properties of the attribute;

(5) fp: $\Gamma \rightarrow S_T$, $S_T \subset S$ is a total function, which shows the properties of the ports;

(6) D: $\Gamma \rightarrow 2^A$ is a total function, for each $g \in \Gamma$, $D(g)$ is the collection of the attributes which can be modified via port g.

Definition 7. Component signature morphism. Given two component signatures $\theta_1 = <\Sigma_1, A_1, \Gamma_1, fa_1, fp_1, D_1>$ and $\theta_2 = <\Sigma_2, A_2, \Gamma_2, fa_2, fp_2, D_2>$, a morphism σ: $\theta_1 \rightarrow \theta_2$ from θ_1 to θ_2 consists of:

(1) a morphism of algebraic signatures σ_v: $\Sigma_1 \rightarrow \Sigma_2$;

(2) for each f: $s_1, \ldots, s_n \rightarrow s$ in A_1, an attribute symbol $\sigma_a(f)$: $\sigma_v(s_1), \ldots, \sigma_v(s_n) \rightarrow \sigma_v(s)$ in A_2;

(3) for each g: $s_1,\ldots,s_n$ in Γ_1, an action symbol $\sigma_\gamma(g)$: $\sigma_\gamma(s_1),\ldots,\sigma_\gamma(s_n)$ in Γ_2;

(4) for each $g \in \Gamma_1$, $\sigma_a(D_1(g))=D_2(\sigma_a(g))$.

The last conditions show that the attributes affected by a certain port must be preserved through a component signature morphism.

Definition 8. Component specification. A component specification CS is a pair (θ, Δ), where θ is a component signature $<\Sigma, A, \Gamma, fa, fp, D>$ and Δ, the body of the specification, is a quadruple (I, F, B, Φ), where

(1) I is a θ-proposition (constraining the initial values of the attributes);

(2) F assigns to every port $g \in \Gamma$ a non-deterministic command, i.e. F maps every attribute a in $D(g)$ to a set expression $F(a)$;

(3) B assigns to every port $g \in \Gamma$ a θ-proposition as its guard.

(4) Φ is a (finite) set of θ-formulae (the axioms of the description), which is a collection of the functional and non-functional goals of the component.

We distinguish between functional requirements and nonfunctional requirements. Functional requirements describe the system behavior as well as the collaboration among system components to accomplish the system behavior. nonfunctional requirements pertain to how a system performs its functions and include concerns such as quality, quantity, and timeliness.

Definition 9. Component specification morphism. A morphism ω: $CS_1 \rightarrow CS_2$ of component specification $CS_1 = <\theta_1, \Delta_1>$ and $CS_2 = <\theta_2, \Delta_2>$, consists of a signature morphism $\sigma: \theta_1 \rightarrow \theta_2$ such that,

(1) $\exists p \in \Phi_1$, $\omega(p) \in \Phi_2$;

(2) $\exists g_1 \in \Gamma_1$, $a_1 \in D_1(g_1)$, $B_2(\sigma(g_1)) \supset \sigma(F_1(g_1, a_1))=F_2(\sigma(g_1), \sigma(a_1))$;

(3) $I_2 \supset \omega(I_1)$.

(4) $\exists g_1 \in \Gamma_1$, $B_2(\sigma(g_1)) \supset \sigma(B_1(g_1))$.

Requirements shown above allow guards to be strengthened but not to be weakened.

3.2 Component Relations and The Hierarchy Component Models

Relationships between components impose accessibility constraints on their attributes and, thus, restrict the way components can be interconnected. In the component-based model-driven development [9], there are many kinds of relations between component models, such as *compose*, *use*, *extend*, as well as the *mapping* relations between component models at different abstract levels [10].

A specification morphism m: $A \rightarrow B$ from a specification A to specification B maps any element of the signature of A to an element of the signature of B that is compatible (i.e., sort with sort etc). The *compose* relationship express how that

component is part of the given ones. On this basis, a *compose* relation between two components S_1 and S_0 is achieved in category theory by identifying a morphisms c_1 from S_1 to S_0, which express that S_1 is a subcomponent of S_0. This case is expressed by Fig 3 (a). Through this morphism, the configuration diagram returns a new component that represents the overall system. Some constraints, however, apply. The *use* and *extends* relationship may describe a dependency between two implementations or between two specifications. It actually applies to different yet closely related component relationships. These dependency relationships between components at the same level are represented by the morphisms given in Fig. 3 (b) and (c), which shows that the implementations of some functions in R_1 (or C_2) are based on the functions specified in R_2 (or C_1). The *mapping* relations describe the key relationship between abstract component specifications and concrete component specifications. It is also formalized via morphisms in category theory. As shown in Fig.3. (d), the morphism between S and T is as an illustration, where T is the direct corresponding part to S at a more concrete level. The *mapping* relationship can be defined informally as follows: *Abstract component S* is mapped to *concrete component T* if and only if *T* exhibits the behavior specified by *S*.

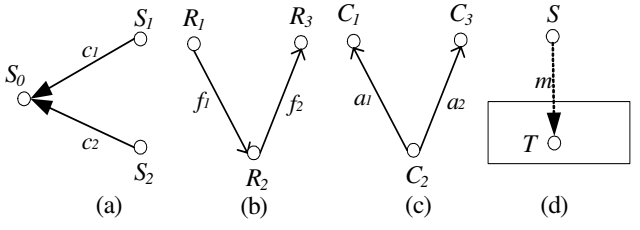


Fig. 3. Component specifications morphisms

The composition of component specifications can be modeled hierarchically in a category theoretic framework. Large complex systems are put together, or configured, from smaller parts, some of which have already been put together from even smaller parts. The composition operation then defines and constructs an aggregated component describing the overall system from the individual components and their interactions. Colimits can be used to construct systems from simpler components in our category of component models. We consider systems composed of a number of components coordinating their activities. The components of a system are represented by recursively defined objects and configured by combinators. Under such interpretation, a categorical diagram represents a system of components. A colimit of a diagram, if it exists, allows one to represent the whole system as a single component.

Properties can be associated to each specification. These are the properties that we expect the component to respect; that we need to prove on the component. We can represent these properties in the same framework as the specifications and this allows us to use category theory and particularly categorical computations to manage them. The property of the colimit specifications gives the composition for the properties. The advantage of this approach is that the management of properties and their status (proved, to be proved) is handled in a uniform way through the management of morphisms and proof obligations.

3.3 Architecture Models and Mapping Functors

Software architecture is a world populated by components, connectors, configurations, etc [6]. As a simple example, an architecture theory could be defined by the objects and the composition rules. These rules provide the semantics of the architecture, and can be used to both interpret the meaning of structures and to identify equivalent or included substructures. These notions can be formalized as a category.

Definition 10. Architecture Model. An architecture model is a 5-tuple $<CO, CR, OT, RT, \vdash>$, where

(1) CO is a collection of components;
(2) CR is a collection of binary relations defined over CO;
(3) OT is a collection of component specifications, and for every component $o \in CO$, $type(o) \in OT$, herein the operation *type* returns the component's type;
(4) RT is a collection of binary relation types defined over OT, for each $r \in CR$, $type(r) \in RT$;
(5) $\vdash$ is a satisfaction relation between OT-sentences and OT-models [10] such that $\vdash$ defines a well order.

Component specifications and cs-morphisms constitute a category for architecture model, henceforth denoted by AM. Obviously, the category AM is cocomplete.

Category theory also provides us with the means to establish relationships between different architectural models: functors. An architecture mapping from AM_1 to AM_2 is simply a mapping of component specifications of AM_1 to component specifications of AM_2 that preserves relations between these components.

Definition 11. Architecture mapping functor. An architecture mapping functor denoted $F: AM_1 \rightarrow AM_2$ from architecture model $AM_1 = <CO_1, CR_1, OT_1, RT_1, \vdash>$ to $AM_2 = < CO_2, CR_2, OT_2, RT_2, \vdash>$ is a function $F: AM_1 \rightarrow AM_2$, in such a way that

(1) for every component $o \in CO_1$, $F(o) \in CO_2$;
(2) for every component specification o, $o' \in CO_1$, $o \rightarrow o' \in CR_1$ implies $F(o) \rightarrow F(o') \in CR_2$;
(3) $F(f \circ g) = F(f) \circ F(g)$; whenever $g, f \in CR_1$ and $f \circ g$ is defined;
(4) for all OT_1 sentence s, $AM_1 \vdash s$ if and only if $AM_2 \vdash F(s)$.

According to the theory of model-driven development [11], a mapping functor between two architecture models at different abstract levels for the same system is a mapping in case the axioms of the source are logically implied by the axioms of the target under the translation. Thus, architecture mapping preserve the properties of the source architecture models. Functors map the objects and morphisms of one category to corresponding objects and morphisms of another category. Consistency between the sorts and operations of the component specifications are maintained.

4 A Case Study

In this section, a component-based model for email client was used as a simple case to illustrate the feasibility of the approach proposed in this paper.

Based on hierarchy component model, the structure of the source model was depicted within categorical diagram in the left part of Fig.4., which involving seven components types: (1) *MainUI* is in charge of the UI layout and art design of the interaction between the mail client and users, through which users can receive email, check email, send email and compose email; (2) *EmailManagement* is responsible for the storage, reading and display of all the e-mails stored locally; (3) *Editor* is used to composing text format or html format e-mail; (4) *Client* is responsible for sending and receiving e-mail; (5) *Protocol* is identified for the setting of mail protocols; (6) *AddressBook* is used for the management of address book; (7) *Account* is responsible for account management. Herein, *EmailManagement* and *Editor* are two composite components. In the component *EmailManagement*, a general-used component named *GeneralList* used to handle mail-lists, a *DataAccess* component used to access email information, and a component *FileView* used to show details of different kinds of e-mails as well as the corresponding component *ManageUI* for user interface were introduced as four sub-components. Herein, the component *FileView* was composed of three sub-components: *HtmlView* to deal with Html format documents, component, *TextView* to manage text format documents and *MultimediaView* to process multimedia documents. In the component *Editor*, the sub-component *EditorUI* is responsible for the user interface, and two subcomponents named *TextEdit* and *HtmlEdit* respectively are used to compose different formats emails.

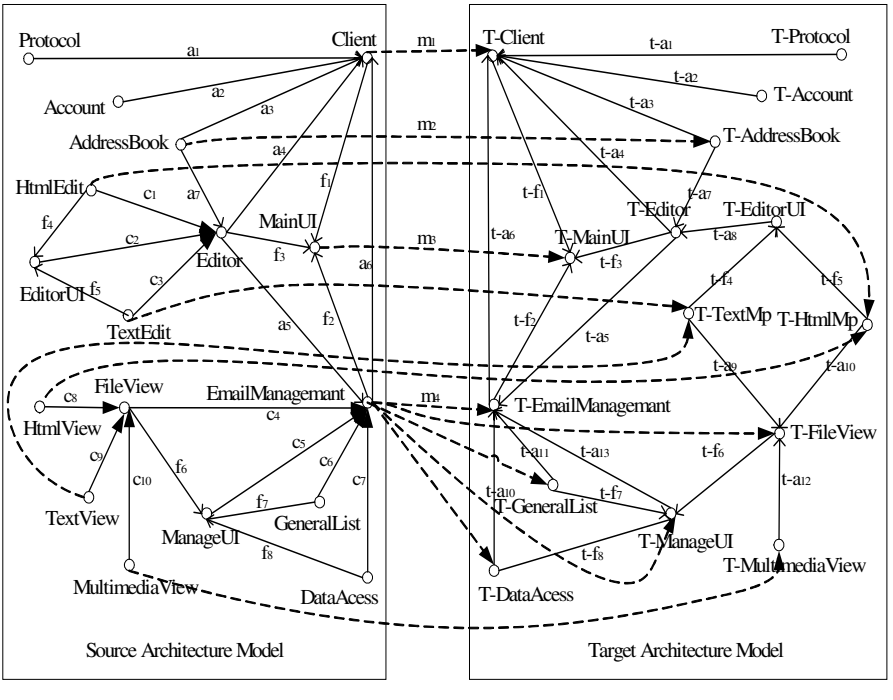


Fig. 4. Component model mapping of email client

We assume that the target platform does not support composite components, such as the programming language C++ does not support nested definition of the Class, and the EJB specification, only permits several javabeans be included in a jar package, but do not support the definition of composite EJB. In such case, the mapping relations between the majority of the source atomic components and the components types of the target platform can easily built. As for the composite components, the decomposition mapping relations must be built through stepwise layers-decomposition. In order to optimize the target architecture, there is generally a need to integrate target components specifications, and thus will form the composition mapping from the source model to the target model. In this case, the functions for html document editing and browsing were combined into a component *T-HtmlMp* in the target architecture. Similarly, the functions for text document editing and browsing were combined into a component *T-TextMp*.

The corresponding target architecture model was represented within categorical diagram in the right part of Fig.4. The mapping relations from the source to the target can be observed by component names. Only a part of mapping morphisms are drawn in Fig. 4, which satisfy the commutative law of category diagram, such as $t\text{-}f_1 \circ m_1 = m_3 \circ f_1$, $tf_3 \circ t\text{-}a_7 \circ m_2 = m_3 \circ f_3 \circ a_7$, and so on. This property shows that the transformation following these mappings persevere consistency of dependency relations among the components.

5 Related Work

In the past few years, a large number of approaches for model transformation have been proposed. Most of these approaches lay emphasis on providing a concrete solution for the transformation from source model to target model. In the work by Bezivin et al [12], the impact on the efforts to define mapping rules caused by the gap between the source and the target modeling languages is mentioned briefly from the view of meta-model semantics, but no general solutions are given. The central role of formalism extension mechanisms in managing the abstraction-level gap between modeling languages as well as the platform-level details of specific implementations is shown in Caplat and Sourrouille's work [3]. The gap between the modeling languages can be narrowed using this mechanism, but cannot be completely eliminated. The mapping relations between models are still difficult to define directly. On the other hand, category theory has been widely used to facilitate specification construction. In Gerken's work [6], category theory and algebraic specifications were used to develop a formal definition of architecture and it also showed how architecture theory can be used in the construction of software specifications. The problem of interconnection relationships in large systems was addressed using category theory in Guo's paper [10], which also gives a framework of the dependencies modeling. The work by Fiadeiro and Maibaum [7] have showed how elementary concepts of category theory can be used to formalize key notions of software architecture independently of the formalism chosen for describing the behavior of components. Despite the popularity of category theory in specification construction, little attention was given to understanding the relationship between levels of abstraction for component-based model mapping.

6 Conclusion and Future Work

In this paper, a unifying framework for component-based model mapping was presented. The framework is based on category theory due to its formality and its high level of abstraction. An important contribution is the formalization of mappings between architecture modes at different abstract levels. In order to specify and verify such a model mapping to ensure semantic compatibility, we postulate that through formality, the terms "component" and "architecture" both can be precisely defined and some important properties of systems can be investigated with precision. Furthermore, we use category theory to develop a formal definition of architecture mapping and some important properties are exploited. It can be a measurement for validating the mapping rules between models at different abstract levels, and thus to provide an effective support to model driven software development.

Future works are as follows: (1) further to formalize the definition of component specification and architecture model, and thus to strengthen the abilities of semantic expressiveness and consistent verification between models; (2) more study about the preserving of semantics features in model mapping for the enhancement of accuracy.

References

1. Brent, H., Peri, T.: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal 45(3), 451–461 (2006)
2. Krzysztof, C., Simon, H.: Feature-based survey of model transformation approaches. IBM Systems Journal 45(3), 621–644 (2006)
3. Caplat, G., Sourrouille, J.L.: Model Mapping Using Formalism Extensions. IEEE Software 22(2), 44–51 (2005)
4. Barr, M., Wells, C.: Category Theory for Computing Scince. Prentice-Hall, Englewood Cliffs (1990)
5. Goguen, J.: A Categorical Manifesto. Mathematical Structures in Computer Science 1(1), 49–67 (1991)
6. Mark, J.G.: Specification of Software Architecture. Journal on Software Engineering and Knowledge Engineering 10(1), 69–95 (2000)
7. Fiadeiro, J.L., Maibaum, T.: A Mathematical Toolbox for the Software Architect. In: Proc. 8th International Workshop on Software Specification and Design, pp. 46–55 (1995)
8. Eilenberg, S., MacLane, S.: General theory of natural equivalences. Transactions of the American Mathematical Society 58(1), 231–245 (1945)
9. Colin, A., Joachim, B., Christian, B., et al.: Component-Based Product Line Engineering with UML, Addison-Wesley Professional, Pearson Education, Boston (2002)
10. Guo, J.: Using category theory to model software component dependencies. In: ECBS 2002. Proc. of the 9th Annual IEEE Int'l Conf. and Workshop on the Engineering of Computer-Based Systems, pp. 185–192. IEEE Computer Society, Los Alamitos (2002)
11. Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley, Boston (2003)
12. Bezivin, J., Hammoudi, S., Lopes, D., Jouault, F.: Applying MDA approach for Web service platform. In: Proc. of Enterprise Distributed Object Computing Conference, Monterey, California, USA, pp. 58–70 (2004)

Exploiting Thread-Level Parallelism of Irregular LDPC Decoder with Simultaneous Multi-threading Technique

Xing Fang[1], Dong Wang[1], and Shuming Chen[2]

[1] School of Computer, National University of Defense Technology, 410073, Changsha, Hunan Province, P.R. China
{fangx8009,nudt_jum}@163.com
[2] School of Computer, National University of Defense Technology, 410073, Changsha, Hunan Province, P.R. China
smchen@nudt.edu.cn

Abstract. Irregular LDPC (Low Density Parity Check) code is a powerful error correction code in wireless communication applications. However, irregular LDPC decoder has limited instruction-level parallelism. This paper exploits the thread-level parallelism of irregular LDPC decoders with simultaneous multi-threading (SMT) techniques. The simulations with random constructed parity check matrixes under different signal-to-noise ratios and three block lengths show that it can attain 16.7%~45.3% performance improvement by SMT technique with the area cost increasing by about 17.73%, which supposes that SMT is an efficient technique to improve the performance of irregular LDPC decoders.

1 Introduction

LDPC codes have caused major attention in the research community in recent years because of their excellent error correction capability and performance [1] [2] [3]. Traditionally, LDPC decoders are implemented with Application Specific Integrated Circuits [4] [5]. However, these implementations lack of flexibility, and require the parity check matrix with regular expressions. With the development of Software Defined Radio and the emerging of new methods to create LDPC codes, it is urgent to research the software implementation of LDPC decoder.

However, exploiting the instruction-level parallelism of LDPC decoders meets two problems. First, irregular LDPC decoders consist of multi-layer loops the branch operations limit the instruction-level parallelism. The uncertain iteration times of inner loops make the thing even worse, and reduce the probability of exploiting instruction-level parallelism through loop-unrolling and software-pipeline. Secondly, the inner loops of LDPC decoders require three Loads/Stores operations for each non-zero operand of sparse matrixes, two of which are typically expressed as indirect subscripts, such as S(i(j)), which makes the function units in the state of waiting for the operand for most of time.

This paper would focus on exploiting thread-level parallelism of irregular LDPC decoders. SMT implements multiple virtual hardware threads on a single chip, and

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 650–657, 2007.

uses the idle function units to increase processors cost efficiency. This paper maps an irregular LDPC decoder onto a SMT digital signal processor (DSP) to exploit thread-level parallelism of irregular LDPC decoders.

The rest of this paper is organized as follows: Section 2 introduces the decoding algorithm; Section 3 introduces the dual-thread implementation of the irregular LDPC decoder; Section 4 gives a brief description of the hardware platform and the architecture parameters; Section 5 gives the prime experimental results and analysis; Section 6 concludes the paper.

2 Description of Irregular LDPC Decoders Algorithm

LDPC codes can be decoded by Gallager algorithm of iterative Two Phase Message Passing. Widely used decoding algorithms include sum of products (SP), min-sum (MS) and Jacobian based BCJR. Compared to these algorithms, the offset min-sum (OMS) decoding algorithm is less computation intensive, and can achieve the same bit-error rate (BER) as that of floating point SP and BCJR with 0.1dB SNR penalty. Since OMS is suited to be implemented on fixed-point DSP [6], we use OMS decoding algorithm in this paper.

In the following description, we use N(m) to denote the symbol-node set associated with the check-node m; M(n) to denote the check-node set associated with the symbol node n; L(xn) to denote a posterior probability for symbol node n; $\lambda_{m,n}$ to denote the log likelihood ratio transferred from a variable node to a check node; $\Lambda_{m,n}$ to denote the log likelihood ratio transferred from a check node to a variable node. An iteration of the algorithm is illustrated in Figure 1 using the Tanner graph associated with the LDPC code, where symbol nodes are shown as circles and check nodes as squares.

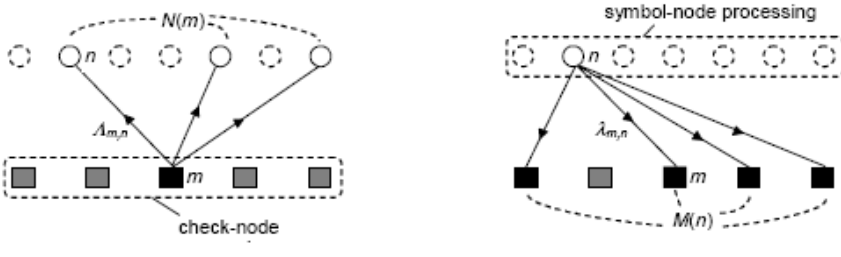


Fig. 1. An iteration of decoding algorithm on a Tanner graph associated with the LDPC code

The decoding process consists of four steps as follows:

1) Initialization

$$\lambda_{m,n} = L(x_n) \tag{1}$$

2) Processing at check nodes

a) For each $n \in N(m)$, compute

$$\phi^{-1}\left(\tilde{\Lambda}_{m,n}\right)=\max\left(\min_{n'\in N(m)\backslash\{n\}}\left|\lambda_{m,n'}\right|-\beta,0\right) \tag{2}$$

b) Compute the LLR:

$$\Lambda_{m,n}=\phi^{-1}\left(\tilde{\Lambda}_{m,n}\right)\prod_{n'\in N(m)\backslash\{n\}}\operatorname{sgn}\left(\lambda_{m,n'}\right) \tag{3}$$

3) Processing at symbol nodes

a) For each $m\in M(n)$, update:

$$\lambda_{m,n}=L(x_n)+\sum_{m'\in M(n)}\Lambda_{m'n}-\Lambda_{m,n}=\lambda_n-\Lambda_{m,n} \tag{4}$$

b) Hard decision

$$\hat{x}_n=(\lambda_n>0)?0:1 \tag{5}$$

4) Export the result or start a new iteration. If $H\hat{x}=0$ or the maximum iteration number is reached, then export the result, else start a new iteration from step 2.

In equation (2), β is a correcting offset equal to a positive constant. The selection of β depends on the code parameter.

3 Multithreaded Implementation

From the last section we can see that, in the same iteration the process on a check node or on a variable node is independent from the process on other nodes. And the termination criteria can also be partitioned into two parts, as shown in equation (6):

$$Hx=0\Rightarrow\begin{bmatrix}H_1\\H_2\end{bmatrix}x=0\Rightarrow\begin{cases}H_1x=0\\ \&\&\\H_2x=0\end{cases} \tag{6}$$

So we can easily divide the algorithm into two threads in term of different nodes sets. The pseudo code of the dual-thread implementation is shown in Fig. 2.

4 Hardware Experiment Platform

Simultaneous multi-threading (SMT) technique [7] is often implemented in many superscalar processors, such as Intel Hyper-threading technique [8]. The MOSI (Multi-operation Splitting Issue) micro-architecture technique proposed in [10] provides a way to implement SMT on very long instruction word (VLIW) processors.

***P**: the maximum iteration* ***N**: the block length* ***M**: the number of check nodes*	
Thread 0	***Thread 1***
for *i =0 to N/2-1* ***for*** *each* $\lambda_{m,i}$ *in the row i* *initialize with equation (1)* ***end*** *calculate* $\hat{x}_i$ *with equation (5)* ***end*** ***barrier ();*** ***for*** *iteration =0 to P* *Calculate stop criteria with equation (6)* ***barrier ();*** *if stop criteria is true* *break;* ***for*** *j =0 to M/2-1* *Calculate equation (2)-(3)* ***end*** ***barrier ();*** ***for*** *k =0 to N/2-1* *Calculate equation (4)-(5)* ***end*** ***barrier();*** ***end***	***for*** *i =N/2 to N-1* ***for*** *each* $\lambda_{m,i}$ *in the row i* *initialize with equation (1)* ***end*** *calculate* $\hat{x}_i$ *with equation (5)* ***end*** ***barrier ();*** ***for*** *iteration =0 to P* *Calculate stop criteria with equation (6)* ***barrier ();*** *if stop criteria is true* *break;* ***for*** *j =M/2 to M-1* *Calculate equation (2)-(3)* ***end*** ***barrier ();*** ***for*** *k =N/2 to N-1* *Calculate equation (4)-(5)* ***end*** ***barrier();*** ***end***

Fig. 2. The pseudo code of dual-thread implementation of irregular LDPC decoder

We implement a SMT digital signal process (DSP) prototype YHFT DSP/900, which consist of two hardware threads and is based on MOSI micro-architecture technique, as depicted in Fig. 3. It consists of eight function units, and can execute up to 8 instructions simultaneously. Except for the instruction fetch buffers, the register files and the write-back buffers, other on-chip resource is shared between two hardware threads.

The fast hardware barrier implemented in YHFT DSP/900 is depicted in Fig. 4. When a hardware thread writes to the memory-mapped synchronization register and if the state of the synchronization register is 0, the synchronization register is changed to 1, and the hardware thread enters waiting state; otherwise the synchronization register is changed to 0, and all waiting hardware threads are released from the waiting state.

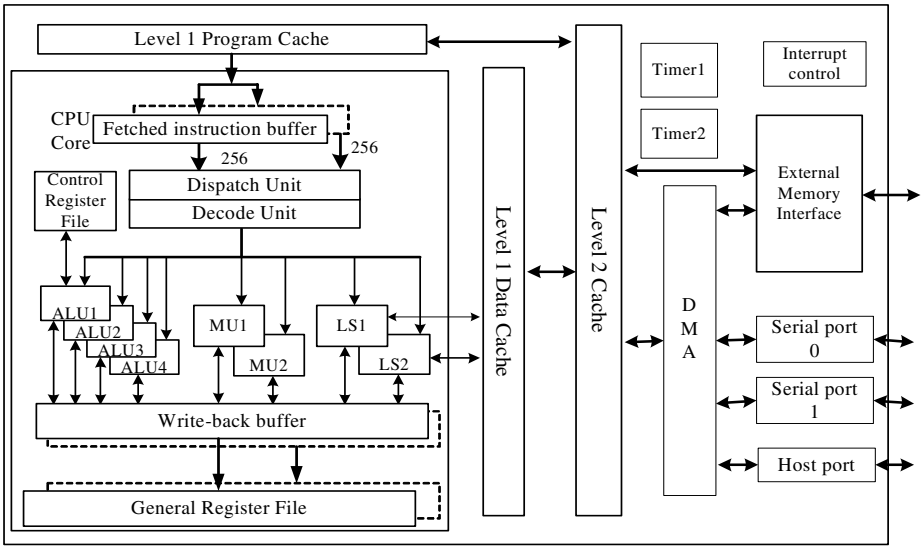


Fig. 3. The architecture block diagram of the SMT DSP: YHFT DSP/900

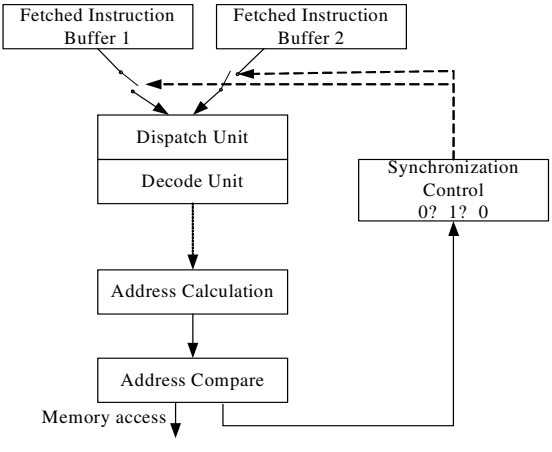


Fig. 4. Hardware barrier mechanism implemented in the SMT DSP: YHFT DSP/900

When a hardware thread enters waiting state, no more instructions of the hardware thread will be dispatched until the thread is released. To ensure the execution correctness, we insert a certain number of NOP instructions after the synchronization operation. The number of NOP instructions depends on the delay from the instruction dispatch to the synchronization. In our implementation, we set this number to be 10.

Table 1 lists main architecture parameters of the hardware platform, in which the access miss latency of L1P cache and L1D cache is known, but the miss latency of L2 cache is influenced by many factors (primary the uncertain access delay of off-chip DRAM memories). In this simulation experiment, we use the average miss latency of L2 cache as the simulation parameter.

Table 1. The main architecture parameters of the hardware platform

Parameters	Values
Clock frequency	200MHz
# of threads	2
Thread priority	Round robin
L1P Cache	4KB 2-way associative/LRU/read allocate
L1P miss	5 Cycles
L1D Cache	4KB/256B/2-way associative /LRU /read allocate
L1D miss	4 Cycles
L2 Cache	64KB/1024B/4-way associative /LRU /read-write allocate/
L2 Cache miss	68 Cycles

5 Experiment Results

5.1 Algorithm Verification

We verify our algorithm by comparing the result with the result of the reference software provided by Radford M. Neal available at [11]. We use sparse matrixes generated randomly as our parity check matrixes, with the check distribution of the parity check matrix is:

$$prop \times count / prop \times count / prop \times count = 0.2x2/0.7x3/0.1x7$$

In which prop is the proportion of symbol nodes associated with count check nodes. In this paper, we use code rate of 1/2, and three block lengths for irregular LDPC codes: N=500, 2500, 10000.

5.2 Performance Analysis

The decoding rate comparison of single-thread irregular LDPC decoders against dual-thread irregular LDPC decoders under three block lengths and different signal-to-noise ratio (SNR) is shown in figure 5. It is shown that the dual-thread decoder attains performance speedup of 47.8%, 17.3% and 16.7% for the block length of 500, 2500 and 10000 respectively.

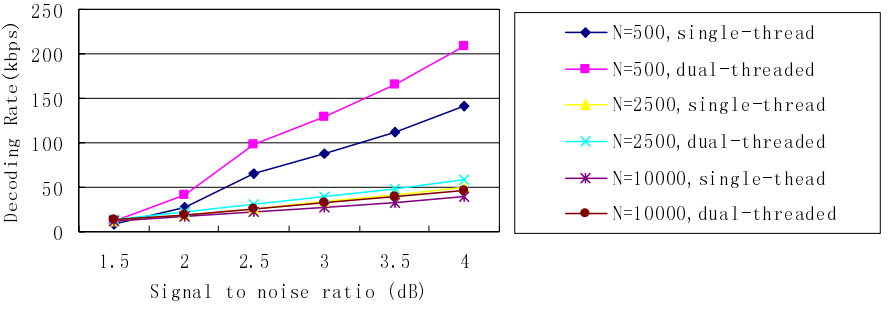


Fig. 5. The decoding rate of irregular LDPC decoders under three block lengths

The decoding rate is influenced by the average iteration time and the execution time of iteration. The average iteration time of different block lengths is shown in figure 6. Thereby, when the channel status is well, we can use smaller block lengths to improve the decoding rate. When the channel status is bad, we can use larger block lengths to get better decoding rate and bit error rate.

The influence of the pipeline stalls caused by cache misses are shown in figure 7. We can see from figure 7 that using simultaneous multi-threading techniques can improve the real execution time by about 43%. But when the block length is large, the total execution time is dominated by pipeline stalls caused by cache misses, which impairs the benefits of simultaneous multi-threading.

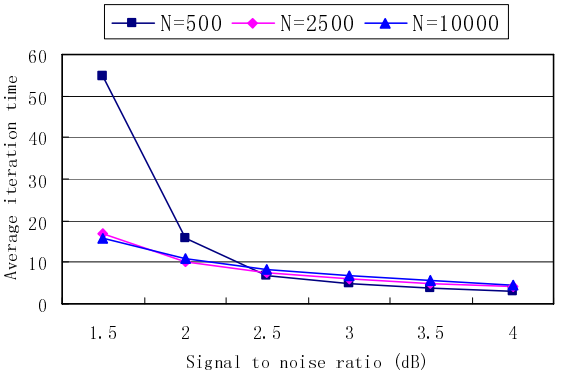


Fig. 6. Average iteration times of irregular LDPC decoder under three block lengths and different SNR

5.3 Cost

Under SIMC 0.18μm process and synthesized by DC of Synopsys®, YHFT-DSP 900 area is about 14,462,947μm2, which increases by about 17.73% than single-thread processor YHFT-DSP/700[9]. The frequency of YHFT-DSP 900 can reach about 250MHz at the best case.

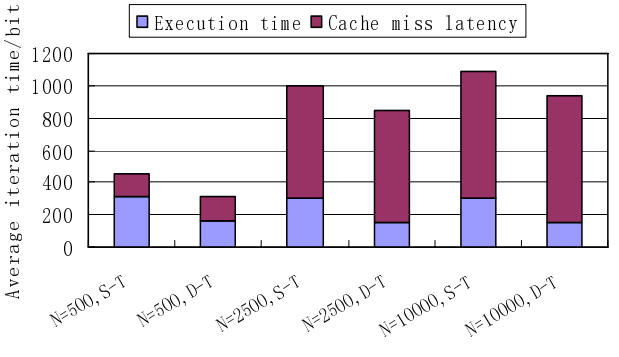


Fig. 7. The impact of cache misses latency on performance of irregular LDPC decoder

6 Conclusion

The experiment results of this paper exhibits that exploiting the thread-level parallelism of irregular LDPC decoders on DSPs based on SMT techniques is an efficient way to improve decoding rate. With different block lengths and signal-to-noise ratio, we attain 16.7% ~ 45.3% decoding rate improvements, while the area cost only increases by about 17.73%.

When the block length is large, long latency caused by cache misses will dominant the total execution time. So in the future research, we will focus on more efficient prefetch strategies to alleviate the impact of cache misses.

Acknowledgement

This work is supported by National Science Foundation of China (60473079).

References

1. Gallager, R.G.: Low Density Parity Check Codes. MIT Press, Cambridge, MA (1963)
2. Gallager, R.G.: Low-density parity-check codes. IRE Transactions on Information Theory IT-8, 21–28 (1962)
3. MacKay, D.J.C.: Good error-correcting codes based on very sparse matrices. IEEE Transactions on Information Theory 45, 399–431 (1999)
4. Karkooti, M., Radosavljevic, P., Cavallaro, J.R.: Configurable, High Throughput, Irregular LDPC Decoder Architecture: Tradeoff Analysis and Implementation. In: ASAP 2006. IEEE International Conference on Application-specific Systems, Architectures and Processors, IEEE Computer Society Press, Los Alamitos (2006)
5. Rovini, F.R.M., L'Insalata, N., Fanucci., L.: VLSI Design of a High-Throughput Multi-Rate Decoder for Structured LDPC Codes. In: Proceedings of the 2005 8th Euromicro conference on Digital System Design, pp. 202–209 (August 2005)
6. Chen, J., Dholakai, A., Eleftheriou, E., Fossorier, M., Hu, X.: Reduced-complexity decoding of LDPC codes. IEEE Transactions on Communications 53, 1288–1299 (2005)
7. Dean, T., Susan, E., Henry, L.: Simultaneous multi-threading: Maximizing on-chip parallelism. In: Proceedings of the 22nd Annual International Symposium on Computer Architecture, Italy, pp. 392–403 (1995)
8. Marr, D.T., Binns, F., Hill, D.L., Hinton, G., Koufaty, D.A., Miller, J.A., Upton, M.: Hyper-threading technology architecture and microarchitecture. Intel Technology Journal 6(1), 4–15 (2002)
9. Chen, S., Li, Z., Wan, J., et al.: Research and Development of High Performance YHFT Digital Signal Processor. Journal of Computer Research and Development (Chinese) 43 (2006)
10. Wan, J., Chen, S.: MOSI: A SMT Microarchitecture Based on VLIW Processor. Chinese Journal of Computer Science 29(3) (March 2006)
11. http://www.cs.utoronto.ca/ radford/LDPC-2006-02-08

P2P Distributed Cooperative Work Model Based on JXTA Platform

Gao Bao-Qing, Fu Xiu-Fen, and Xu Su-Xia

School of Computer, Guangdong University of Technology, Guangzhou 510075, China

Abstract. Distributed cooperation work system performs more effectively than a centralized cooperation work system in many aspects. In this paper, we analyze the existing problems in cooperation work models nowadays, and then present a P2P distributed cooperation work model based on JXTA platform, which can greatly simplify the development of distributed collaboration work system and enable the system possess of better scalability, platform- -independence and flexible applicability. At last, we obtain an analysis compared to other models and make suggestions to improve the model.

Keywords: CSCW JXTA P2P Distributed Collaboration Concurrency Access Control.

1 Introduction

P2P (peer to peer) is becoming a research hotshot in recent years because of its characteristics such as load balancing, robustness and self-organization, etc. Especially in the CSCW, it gives us a more attentive selection than Client/Server and traditional distributive architecture. Nowadays some business applications based on p2p have appeared in market, including Skype and Microsoft Office Tool Groove. The cooperative application based on p2p can satisfy various cooperation demands with a more natural and cute manner, which is especially suitable for mobile office environment. On the basis of analysing traditional CSCW model, this paper obtains a distributive CSCW model based on JXTA and makes detailed discussion for its layers and working theory, more detailed analyses to the policies used in the model at last.

2 CSCW Model and Existing Problems

CSCW[1] is computer supported coopera- -tive work, which means that exhausting computer and communication technology to support geographically dispersed workers to cooperate in a common task. CSCW system can be divided into two classes: centralized and distributive architecture (also called all coping architecture).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 658–665, 2007.

2.1 CSCW Model

In fact, centralized architecture means Client/Server (or Browser), as shown in Fig1, this architecture is generally composed of one or more servers and more clients interacting with servers, at the same time, which transforms operation events produced by users into instructions that can be recognized by servers and sent to servers to be executed in a unified fashion. At last, servers dispatch and execute these operation instructions, then return processed results to clients to display. Because all processes are centralized in one or a small quantity of servers, collaboration management and Real-time cooperation's concurrent access control can be easily realized

In distributive cooperation architecture, the original functions of servers are balanced to more clients, so that every client has double identity, which enable free interactions between clients (peers) to manipulate their common collaborative task. Every client (peer) only receive cooperation events and execute them in local machine, so the entire collaborative application will gain more rapid responses, but concurrency access control will be more complicated.

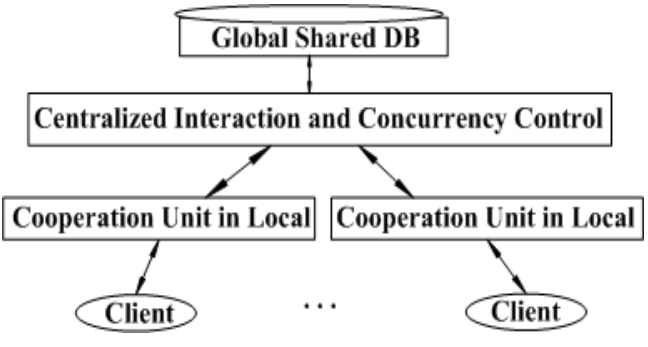


Fig. 1. Centalized CSCW architecture

2.2 Existing Problems in CSCW Model

So far, most of CSCW application systems are based on Client/Server (or browser) [2], such as Microsoft Exchange, IBM Lotus Notes and etc, but the fault of this architecture is obvious. For example, when transferring a mass of multi-media data, the performance of server may be the bottle-neck of a system. The display of Clients' collaborative operation result will be lagging in the environment severely requiring real-time interaction where shared operated object is located in a server. Distributed cooperative architecture is able to support load balancing and get better responses effect. On the other hand, data redundancy in p2p distributed system can avoid single point failure to ensure robustness.

P2P network is a kind of typical DCE (Distributed Computing Environment). Peers, PeerGroup and other conceptions defined in JXTA especially accord with essential characters of CSCW. For example, communication between peers

corresponds to collaboration between group members and PeerGroup to work-group in realistic life. The collaboration work based on P2P solves the problem of Architecture Mismatch [3] between traditional cooperation architecture and cooperation view provided to users.

3 JXTA Technology

3.1 Simple Instruction About JXTA

JXTA is a new technology proposed by SUN Microsystem according to the fault of current p2p system. JXTA mainly supplies fundamental services needed in p2p application in order to establish a general p2p distributed computing platform, so advanced p2p distributed service and application can be built in a simple but effective manner. JXTA adopts two layered topology network using super node frame, through its core p2p protocols, by which a virtual cross-platform ad-hoc network can be easily built over current physical network facilities, as shown in Fig.2.

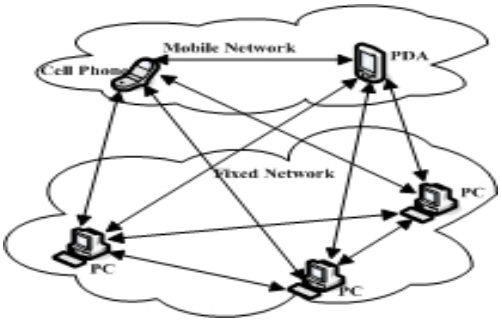


Fig. 2. JXTA virtual all connected network

3.2 Key Conceptions in Developing p2p Distributed Application Using JXTA

JXTA IDs: In the JXTA address mapping model, every resource in network has a unique JXTA ID which is generated while a peer is built. In p2p network, cooperation service and shared operated object are recognized and searched by JXTA IDs.

Peers: A peer may be any entity which can interpret core protocols defined in JXTA. Peer can be thought as an equal priority collaborator.

Messages: Message is a basic unit of transferring data in communication between peers. Peers interact with each other by sending and receiving messages. Two formatted messages, XML and binary messages, are supported by JXTA, developer can make a choice according to different requirements.

Peer Groups: Peer Group is a self-organized team composed of some peers by common interest or a common collaborative task. A Peer Group is actually a dynamic peer set which can penetrate firewall and NAT (Network Address Translator) to support communications between heterogeneous networks, as shown in Fig.2.

Pipes: Pipe is a kind of virtual communicating channel between JXTA service and application. Two peers are connected by a pipe, in which peers' data stream flows. JXTA supports different kinds of pipes to fulfill various p2p communication scenes.

4 P2P Distributed CSCW Model Based on JXTA

This paper proposes a CSCW model based on JXTA, which is used to describe the whole structure and functional layers, as shown in Fig.3. In the model, fixed computing facilities are configured with J2SE JXTA and small scale or mobile devices configured with J2ME JXTA. In the next section, the paper will make an instruction about every layer and detailed analyses on concurrency access control unit used in the model.

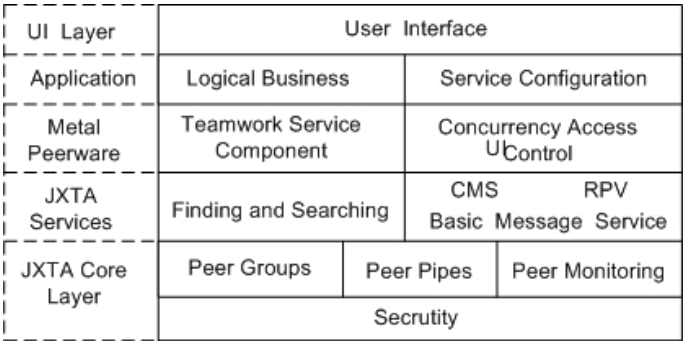


Fig. 3. CSCW model based on JXTA

1) The last two layers are JXTA core protocol layer and fundamental p2p service layer, which provide many important bottom services including peer monitoring, peer finding, shared resource distributed index and etc. In p2p network, every peer dynamically enters or exits and peers aren't able to identify remote peer (mostly because of NAT), so safe problem is outstanding comparatively. JXTA provides a relatively perfect safe frame including network security include user login based on account and password, the authentication infrastructure, simple visit control, encryption and TSL/SSL. Especially for peers running on J2SE or J2EE which support a more perfect security frame.

2) The third layer is composed of groupware[1] components and concurrency access control module. It is not geared to the needs of concrete application, but used to supply some universal cooperation functions like concurrency access control, priority control and interactions between sites that are implemented as fundamental services, so developers make use of this developing concrete cooperation application. Teamwork

Services Components are fundamental elements to realize collaboration such as Publish/Subscribe, file sharing and exploring and asynchronous message service. The concurrency control module is specifically designed for the deficiency of JXTA in the field of real-time coordination; concrete comments will be stated in next section.

3) The fourth is application tier, which includes business logic and service configuration module. Logical business layer is implemented by Java Bean, which mainly contains a business logic adapter. According to different cooperation models (including dialogue model, process model, activity model etc.) ,users can use service configurator to set related groupware components to support changing cooperating requirement, making business implementation without considering details of cooperation implementation.

4) The toppest tier is user interface adapter which is mainly used to provide users with visual service interface. Considering that, when users play PC or other mobile devices their computing capability will be distinctly different, so this tier provides different interface visualization API according to the type of device.

4.1 Model Feasibility and Characteristic Analysis

1) Using JXTA as a fundamental platform in developing p2p distributed application, on the one hand, developers can shake off the complicated job building the frame from the beginning and have more time to be absorbed in developing all kinds of novel and creative applications; On the other hand, the combination of JAVA and XML enables JXTA to be powerful enough to make the interaction between vertical applications. At the same time, to solve the digital producer copyright problem that gravely restricts the development of p2p, JXTA packages the supporting copyright administration and the core function used for centralized peer management, which makes the developing of successful business application become possible.

2) Benefit from using layered structure, when requirements dynamically change, what we need to do is only to revise business realization, because that groupware layer only cares for realization of p2p cooperating in bottom tier, the business logic care for business realization only. Also, when there is any change during the realization of cooperating model, it will not affect the above realization of business logic. By this design, to some extent, most developed component can be reused.

3) The P2P technology is famous for Napster sharing music, at present most P2P applications are confined to the sharing of documents basically. The design of JXTA is also affected by this idea more or less, so that, JXTA does not provide a comparatively perfect concurrency access control mechanism of real-time collaboration. Therefore, the fourth tier of figure3 has added a concurrency control module specially.

At present, most centralized CSCW models mainly use the traditional concurrency control policies [2] which mainly includes two kinds: (1) serial method; (2) lock method including optimistic lock and pessimistic lock; Because traditional methods come from distributed database concurrency control algorithm which does not consider interaction of user interface and only are designed for no-interactive transaction process system, so if the policy is used in groupware system, unnatural Human- --computer interaction, Machine-Machine interaction will be brought on inevitably, and it will block the user interface also.

4) According to above-mentioned problems, Ellis.etc proposed a new kind of concurrency access control method based on Operation- --Translation in 1989. In this algorithm, all operations are defined to a set of partial order relations. For two any operations, such as A and B, if they are interdependent in term of a causal relationship and A keeps ahead B, then, they have this partial order A->B and vice versa. Disordered concurrent operations are allowed, but corresponding translation is necessary according to the relation between operations in order to keep the consistency of sharing operation object in different sites. But in actual application, sometimes, it's very difficult to get all translation rules between all operations. So, this method is not suitable for all circumstances.

The model adopts a new kind of tree-based concurrency control policy [5], every peer maintains current shared operation object by this tree-structure in local. In fact, this tree is document object model DOM structure (Fig.4), any programming language can use a corresponding XML interpreter to transform a application document into this form, this method of work is feasible therefore.

In the process of real-time coordination, every site (peer) preserves the times of operation (logic clock [4]) handled in local. When receiving operation messages from other sites, the site uses its local logic clock to make a compare with the logic lock of the sending messages from source site. if the former is not smaller than the latter, then we think that logical concurrency operations mentioned in last section has occurred, now, the corresponding arbitration module generates a total order of concurrency operation events according to these events' priority, at the same time, the total order message with higher priority is sent to other sites to ensure that operations are executed in the same order in every site, thereby, entire concurrent operation control is achieved.

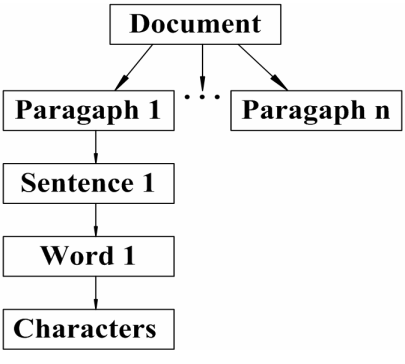


Fig. 4. Tree-structured document model

In this concurrency access control policy, the shared operation object is presented in a tree-structured form, so it has a common adaptability and can control the concurrent granularity freely. By doing this, using this method, the system has more rapid responses and better interactive nature. Currently, using this model, a simple cooperative text editor has been implemented, as shown in Fig.5, in order to validate

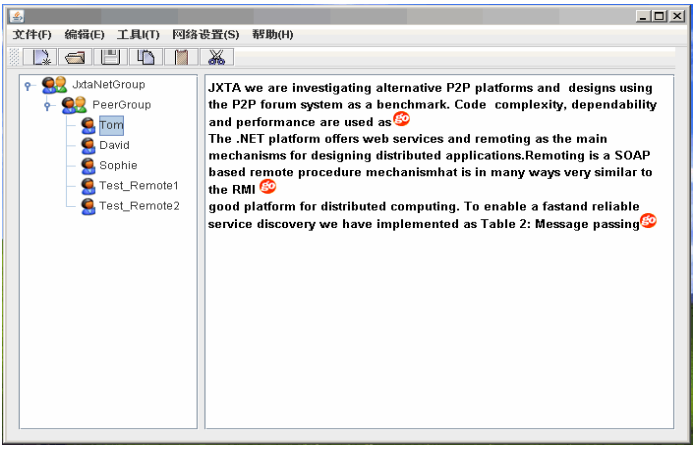


Fig. 5. p2p cooperative text editor

the applicability and performance while handling complicated interactions, a coordination image editor system will be built next step.

5) Verification of model's function and applicability: In order to verify the model's response time and its applicability when network scale is increasing, experiment has measured peers' average response time under the condition that network works on different flow load and has different total nodes. Test configuration parameters as shown in table.1, in which different peers may run on. the same host.

Table 1. Configuration Parameters

Ordinary peer	Pentium 4 2.0G, 512M Memory OS: Windows XP sp2
Rendezvous peer	Celeron 1.7G, 512M Memory Linux 2.4.1s/Federa-Linux 5.0
JXTA Build(all pees)	JXTA 2.4.1
JDK Build(All peers)	JDK 1.6 Hotspot VM

Considering that network scale is limited, only one rendezvous peer is configured. According to the experiment data, we got Fig.6. We can see that when message sending (simulating collaborating operation) frequency declines and the network flow load decreases, peers' response time is shorten. At the same time, the whole performance of the system upgrades as the increasing of network scale (by adding nodes). This is just one of the most different characteristics from Client/Server architecture, and that's the priority of P2P.

6) Existed problems with the model: Every peer is equal to another peer in role, the mechanism work better for peer cooperation though it may be not suitable for

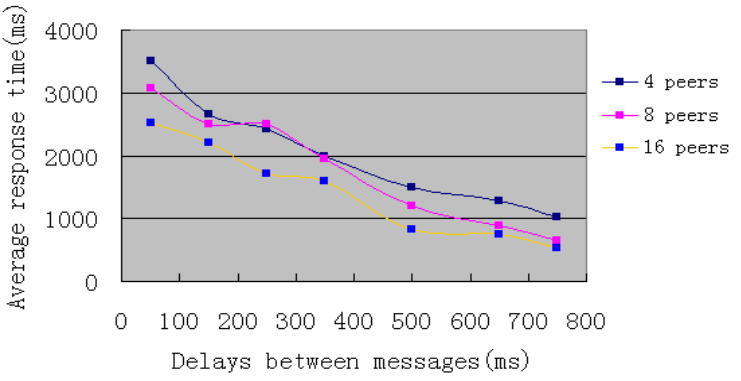


Fig. 6. Response time comparison

cooperation with relation of dependence. So some corresponding improvement to this problem is necessary. For example, the management mechanism based on role will be considered to be added to enhance the applicability of the model. Moreover, the P2P network topology is dynamically changing sharply, which may confront with problems that routine software system does not have. All of this need deeper researcher in future.

5 Conclusions

By the analysis of existed problems in current CSCW system and JXTA's advantage in developing p2p distributed CSCW system, we propose a CSCW model based JXTA, by which many bottom-tier communication details developers confront are shielded. Besides, according to deficiency in handling real-time cooperation, concurrency access control unit based tree-structure is led into the model to get more rapid user interface responses. Research priority next step is: Improving JXTA network's efficiency in data transmitting, makes the model be able to have certain error detecting and error correction ability during the period when exception happens in collaboration.

References

1. Shi, M., Xiang, Y., Yang, G.: The theory and application of CSCW[M], Publishing House of Electronics Industry, Beijing (2000)
2. Hauswirth, M., Podnar, I.: On P2P Collaboration Infrastructures. Infrastructure for Collaborative Enterprise. In: 14th IEEE International Workshops, pp. 66–71 (2005)
3. di Milano, P., da Vinci, P.L.: Peer-to-Peer for Collaborative Applications. In: Proc. Of IEEE Distributed Computing Systems Workshop, pp. 359–364 (2002)
4. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communication of the ACM 21(7), 558–565 (1978)
5. Ionescu, M., Marsic, I.: An Arbitration Scheme for Concurrency Control in Distributed Groupware,[C]. In: An ACM CSCW 2000 Workshop. Proceedings of The Second International Workshop on Collaborative Editing Systems, Philadelphia, PA, vol. 12(3), pp. 329–350 (December 2003)

EasyPAB: An Extensible IDE Framework for Parallel Applications

Yu Ce[1], Sun Jizhou[1], Huang Yanyan[2], Wu Huabei[1],
Xu Zhen[1], and Sun Min[1]

[1] School of Computer Science&Technology, Tianjin University, Tianjin 300072, China
[2] Network Center, Hebei University of Technology, Tianjin 300130, China
yuce@tju.edu.cn

Abstract. Modeling and programming parallel applications are becoming unavoidable for next generation of software architects and programmers, owing to the popularization of multi-core processors and Linux Clusters for high performance/availability computing. As an aid to design and development of various parallel applications running on different parallel computing infrastructure, an extensible IDE (Integrated Development Environment) framework named EasyPAB (Easy Parallel Application Builder) is introduced in this paper. It combines the principle and technology of parallel design patterns and architectural skeleton to simplify the design and development of parallel applications and supports both message-passing-based and shared-memory-based platforms, by providing a unified user interface for modeling visually while generating different types of code skeletons according to specific runtime environment automatically. The implementation of EasyPAB is based on plug-in architecture which is compatible with Eclipse, thus third parties are free to supplement or improve it.

Keywords: Parallel Computing, Design Pattern, Architectural Skeleton, EasyPAB, Eclipse, IDE.

1 Introduction

The development of methodology and tools of software engineering for parallel computing is far behind that of hardware technology. The industry tendency shows that most personal computers (both desktops and laptops) and servers in the future will be equipped with multi-core or many-core processor(s)[KT1], meaning that every computer will be ready for parallel computing. And as for high end market, Linux Clusters are adopted by more and more commercial and research organizations. Any program, which is designed to run on such parallel computing platforms and to take full advantage of the computing capability, has to be parallelized. Unfortunately, parallel programming is so difficult for most programmers who only have experiences in traditional serial programming.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 666–675, 2007.

Aiming at different aspects of parallel programming, types of programming models, specifications, libraries and tools have been brought forward and implemented, such as MPI and PVM (for explicit message-passing programming), OpenMP [CM1] (for multi-threaded shared memory programming), HPF (higher level distributed memory programming), etc. General theories, such as parallel design pattern and architectural skeleton [AG1] [DA1] [WL1] technology, have also been widely discussed. Several platforms or framework for the development of parallel applications [CG1] [HJ1] were designed basing on the above technologies and libraries, such as VisualLinda [KT1], Visper [SK1], SWARM [DV1], etc.

Indeed, paradigms mentioned above have relieved the burden of programmer for a certain degree, yet there is no practical IDE (Integrated Development Environment) for both architects and programmers to build parallel applications. In this paper, we introduces an extensible IDE framework named EasyPAB (Easy Parallel Application Builder). It is based on Eclipse [Ge1], an open source platform on which users can build their own IDE for any kind of application.

EasyPAB is an extensible IDE framework designed for both message-passing-based and shared-memory-based parallel computing infrastructures, such as Cluster and SMP, and supports kinds of parallel programming models, such as MPI, PVM, OpenMP and multi-thread. For higher level of design and development, EasyPAB merges technologies of parallel design pattern and architectural skeleton. The features above are organized and implemented as individual plug-ins that can be added dynamically and become part of EasyPAB. This plug-in-based architecture is consistent with Eclipse. The user interface of EasyPAB is inherited from the workbench of Eclipse, so its layout and most operations are very similar with popular IDEs such as IBM RAD or Microsoft Visual Studio. With the help of EasyPAB, architects or developers can focus on the higher level abstract and solution of applications and needn't care of the complexity of traditional parallel computing technologies.

The rest of the paper is organized as follows. Section 2 reviews and analyzes several essential considerations during the development of parallel applications that EasyPAB concerns. Section 3 describes the architecture of EasyPAB by introducing its three layers and corresponding components. Section 4 discusses the implementation of EasyPAB, shows the plug-ins for visual modeling and source code editor. Section 5 gives the summary and future plan.

2 Modeling and Programming Parallel Apps

When modeling and programming a parallel application, the following preconditions will be taken into account: the features of infrastructure on which it will run; existing libraries or tools which can be utilized; advanced patterns, models and methodologies which are suitable with the special applications. Traditionally, these questions have to be answered one by one, and only experienced developers can master them all. The design of EasyPAB is to provide an integrated environment so that all the works can be easily done as a whole.

Runtime Environment – Parallel Computing Infrastructure
Many types of parallel computing environments have been widely used, including SMP, Cluster, PVP, DSM, MPP, etc. Computers shipped with modern multi-core processor(s) can also be regarded as SMP. As for the design and developing of parallel applications, the most important feature of parallel computing infrastructure is data exchange mode between tasks. From this point of view, parallel computing infrastructure can be classified into two categories: Shared Memory based, such as SMP and DSM, or Message Passing based, such as Cluster and MPP. EasyPAB is designed to support both two types of parallel computing infrastructure and serve the architects and developers with a unified user interface.

Usable Middleware – Parallel Programming Libraries and Tools
The most significant difference between parallel applications and serial applications is that the former have multi-process or multi-thread. Namely, there will be more than one task, which belong to same application and have to communicate with each other, running concurrently. Communication between tasks is the base of any parallel application, while the implementation and management of it are very complex. So some organizations designed and provided specifications and libraries to simplify the work on communications, among them, OpenMP, PVM and MPI are most widely used.

OpenMP is designed for shared memory systems, and it is just a specification, everyone may implement his own library. PVM and MPI are designed for message passing systems. As for most applications, these libraries are sufficient. EasyPAB utilizes them directly instead of developing fire-new ones. To hide the difference among the heterogeneous runtime environments, EasyPAB includes a middle layer encapsulating these existing parallel programming libraries.

General Modeling System - Parallel Design Patterns
Design pattern is used to describe recurring problems and the reusable solutions to each problem. The concept of a design pattern for parallel programming [SM1] is based on the realization that a large number of parallel applications (especially medium- and coarse-grained applications) are built using commonly occurring parallel techniques such as a task-farm or a divide and conquer structure. Parallel design patterns are methodologies, and only experts with rich experiences can use them properly. Besides, parallel design patterns are more useful in analyzing and modeling phase of complex applications than in implementation phase. EasyPAB employs mature parallel design patterns to guide the modeling of parallel applications, and cooperate with other technologies such as architectural skeleton to generate the code framework.

Development Assistor – Parallel Architectural Skeletons
The technology of Parallel Architectural Skeletons is an approach of realizing and using parallel design patterns [WL1]. A parallel architectural skeleton is a set of attributes that encapsulate the structure and the behavior of a parallel design pattern in an application independent manner. These attributes are generic for all patterns. Many of these attributes are parameterized where the value of a parameter depends on the needs of an application. Some of these parameters

are statically configurable while the others are dynamic. User extends a skeleton by specifying the application-dependent static parameters, as needed by the application at hand. Parallel architectural skeleton technology acts as the base of code generation function of EasyPAB.

3 Architecture of EasyPAB

EasyPAB is designed to provide a unified interface for building parallel applications using different programming models on different parallel computing platforms, to help the developers model the applications and generate code framework and necessary runtime configurations. Logically, the architecture of EasyPAB is partitioned into four layers, and the components included in each layer and their dependencies are shown in Fig. 1.

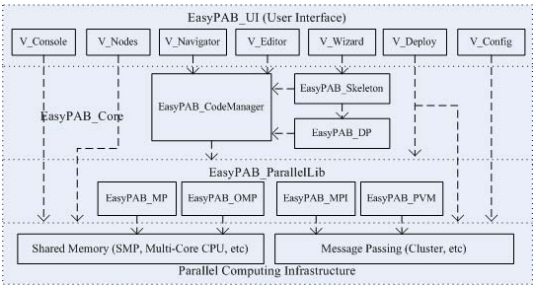


Fig. 1. Architecture of EasyPAB

In the bottom layer, the support for each specific parallel computing infrastructure is implemented as a plug-in with the runtime configuration files and executable scripts which can call the functions of the management systems.

EasyPAB_ParallelLib consists of modules which package different types of parallel libraries. Depending on different types of parallel computing infrastructure, it provides support for MPI, PVM and OpenMP respectively, and provides a unified interface for the upper layer. EasyPAB_MP is designed to support explicit multithread programming, by providing a component of lock-free data structures which allow two or more threads can access the same shared variable without caring of accessing conflict.

EasyPAB_Core is the core layer of the framework. Logically, it is divided into three parts: EasyPAB_CodeManager, EasyPAB_Skeleton and EasyPAB_DP. EasyPAB_DP stores and manages the description of design patterns for parallel applications. On one hand, the abstract definition of solution of each design pattern is used to generate source code skeleton; on the other hand, the comments and examples of each design pattern may be referred by advanced developers. As mentioned above, all the information is stored using a new XML-based language which designed specially for this framework. EasyPAB_Skeleton assists the generation of code for parallel applications. Cooperating with a wizard component

named V_Wizard from the upper layer, this component can generate skeleton of code according to the selected design pattern and customization of developer. The storage of information of all skeletons is same as EasyPAB_DP. Fig.2 shows the details of code generation process.

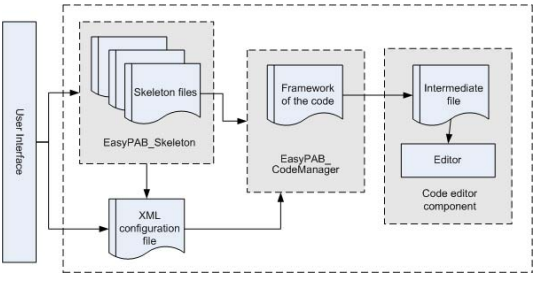


Fig. 2. Generation of Source Code Skeleton

EasyPAB_CodeManager manages the source code generated by EasyPAB and modified by developers. During the initial process of design and development of each application, user must specify the runtime environment and parallel programming library he plans to use. According to those choices, this component will maintain corresponding type of codes automatically.

EasyPAB_UI is the user interface of EasyPAB, and it inherits the general layout and most common functional feathers of Eclipse, so users will feel familiar with the IDE. Besides, it has its own feathers designed specially for building parallel applications, including the support for visual modeling and distributed deployment. Main views of EasyPAB are listed as the following:

V_Wizard helps architects or developers create new projects for parallel applications. It provides a series of dialog boxes through which user can specify the structure of the specific application, and EasyPAB will generate code skeletons automatically.

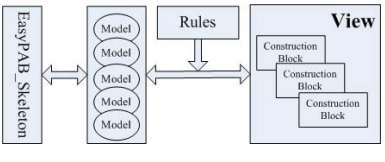


Fig. 3. Visual Modeling System

Fig.3 shows the modeling system of EasyPAB, and V_Wizard acts the View in the figure. The modeling system is based on MVC architecture, where the models are descriptions of specific algorithms or reusable modules, acting as the up layer of EasyPAB_Skeleton. The models are displayed to developers in the View in forms of sets of Construction Blocks which are organized following the Rules

predefined. EasyPAB defined three types of models, namely programming model, process model and communication model. The GUI of modeling system is shown in next section.

V_Console gathers and displays the information during the process of parallel programming, such as output of parallel program compiling, debugging or parallel task deployment. The information could help users to know the details of the program development and make it easier to find the solving method if there are problems existing.

V_Nodes manages the computing resource in the parallel computing infrastructure. It will gather and display the information of each available computing node for user using. User could select the available hosts to use in the view. Since the status of computing resource is dynamic, the information in this view is changed accordingly.

V_Navigator manages the resource of parallel application project which includes source code, executive files and configuration files. It supports resource creation, modification, deletion, orientation and search. Through the Navigator View, user could get a general picture of the project resource.

V_Editor is the area in which user can edit the source code or the other files in the parallel project. In addition to providing all the functionality of an editor, it also supports the key words syntax highlighting and content assistance, this helps user to develop the program quickly.

V_Deploy deploys the binaries of some compiled application to physical computing nodes. This view helps the user to complete the parallel task deployment automatically. The process of deployment depends on the user defined deployment configuration.

V_Config is a visual interface to configure the runtime environment which includes available host configuration, deployment configuration and parallel architecture parameter configuration.

4 Implementation

EasyPAB is not implemented from scratch, but built on an open source platform - Eclipse[Ge1], contributed by IBM. Eclipse is an extensible platform for building IDEs and provides a core of services for controlling a set of tools working together to support programming tasks. Tool builders contribute to the Eclipse platform by wrapping their tools in pluggable components, called Eclipse plug-ins, which conform to Eclipse's plug-in contract.

As the above figure shows, each component of EasyPAB is implemented as a plug-in which is named following the architecture of EasyPAB mentioned in previous section. In EasyPAB_ParallelLib layer, EasyPAB_MP plug-in includes lock-free data structures for explicated multithread programming, and EasyPAB_OMP, EasyPAB_MPI and EasyPAB_PVM plug-ins encapsulate existing implementation of OpenMP, MPI and PVM respectively. Each component in EasyPAB_Core layer is implemented as an individual plug-in with the same

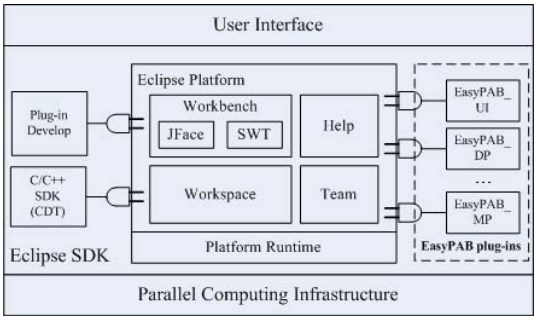


Fig. 4. EasyPAB Plug-ins basing on Eclipse SDK

name shown in Fig.1. All the components in EasyPAB_UI layer are integrated into one plug-in, and each component is implemented as a view.

EasyPAB_CodeManager plug-in provides methods to generate the source code skeleton of the parallel appliction automatically based on the user specified parameters of parallel architectural skeleton. EasyPAB_Skeleton plug-in has realized several typical parallel architectural skeletons which can be used directly. Each parallel architectural skeleton is saved as XML format and has unique key word to be distinguished from each other. It describes the characteristic and the range of usage of the skeleton. It reflects the framework of the parallel strategy and realizes the code of the parallel skeleton. When EasyPAB starts, it will search and load the EasyPAB_Skeleton plug-in dynamically. The unique key word of each parallel architectural skeleton will be extracted and shown in the user interface to help the user to search or select the skeleton. The dynamic loading process is benefit to add or update the parallel architectural skeleton in the library. In this way, user could select the proper parallel architectural skeleton from the library based on the specific applications and specify the parameters of the parallel architectural skeleton, EasyPAB_CodeManager plug-in

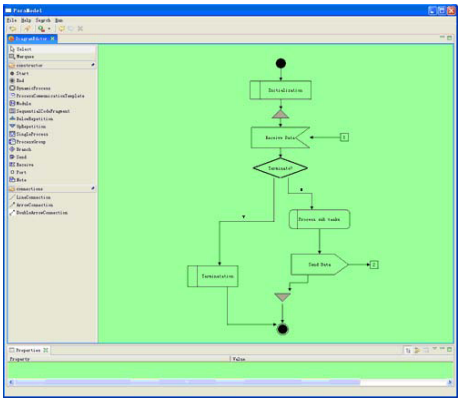


Fig. 5. User Interface for Visual Modeling

will generate the source code skeleton automatically. It simplifies the process of parallel programming and saves the time for the developer.

The plug-in shown in Fig.5 is named ParaModel and acts the user interface of modeling system of EasyPAB, and it depends on EasyPAB_Skeleton and EasyPAB_CodeManager plug-ins. Users can model their parallel application using this plug-in by reusing existed solutions provided by EasyPAB_Skeleton plug-in. When modeling is accomplished, EasyPAB will generate the skeleton of source code according to specific runtime environment automatically with the help of EasyPAB_CodeManager plug-in. The models constructed are saved as XML files.

The model shown in Fig.5 is a programming model, which is suitable for description of specific algorithms or the behavior inside one process. While the other two types of model, process model and communication model, are designed to illustrate the relationship between different process and their communication activities, respectively. Each type of model has its own perspective.

There are four sets of Construction Blocks defined in EasyPAB: structure, action, control, and message. The structure construction blocks are used to describe process, process group, port, etc. The action construction blocks are designed for description of communication and dynamic process structure. The control construction blocks are used to show the control flow of the application or algorithm, such as circulation, conditional branch, or sequence. The message construction blocks are the entities which express the messages interchanged between different processes or process groups, mostly in form of arrows.

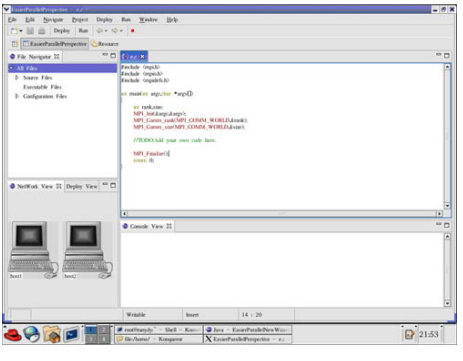


Fig. 6. User Interface for Source Code Edition

Fig.6 shows the source code editor plug-in, and here the runtime configuration is MPI applications running on Linux clusters. After modeling and generation of source code framework, developers can use this plug-in to modify the source code to implement their special applications.

EasyPAB has good expansibility owing to the configurable characteristics which focus on the realization of the data interface. The system uses configuration files to set the runtime environment. It provides three kinds of configuration files: available host configuration file which records the available host in

the parallel computing environment, deployment configuration file which saves the information about parallel task deployment, and parallel architecture parameter configuration file which includes the parameters of the specific parallel architectural skeleton. All of these files are described as XML format which can be parsed easily. Through the configuration files, the system could separate the program abstract model and the part of code realization.

5 Conclusion and Future Work

EasyPAB provides an extensible IDE framework for modeling and programming parallel applications. The implementation is based on Eclipse, a powerful open source platform for customizing special IDEs. With the help of EasyPAB, developers can model the application visually using the modeling system, then the graphs of model will be translated into parallelized code skeleton automatically. The code skeleton is generated and organized according to the design patterns and parallel architectural skeletons which fit that application. After the developers fill in the necessary functional or computing code segments, EasyPAB will compile and deploy the executable binaries according to specific runtime environment. Tested runtime environment including IBM Cluster1350 (a suit of Linux Cluster), and single servers equipped with Intel/AMD dual-core processors (Windows/Red Hat Linux).

The primary characters and innovations of EasyPAB include its extensible architecture, layered abstraction of parallel applications and algorithms, visual modeling technology, and implementation mechanism. The architecture of EasyPAB is extensible so that not only it can be developed incrementally, but also other developer can contribute their own features to it. The layered abstraction is based on the technologies of design patterns for parallel application and parallel architectural skeletons, and supporting automatic generation of source code framework. The visual modeling system provides reusable modules to developers and helps them reuse existing solutions or construct their own applications. The implement of EasyPAB is based on Eclipse, and open source project, so everyone can improve it with new programming models, design patterns, parallel architectural skeletons, as well as support for new parallel computing infrastructures. There are two ways to extend EasyPAB, one is to enhance the functionality of the IDE basing on the Eclipse extension points, and the other is to develop a new plug-in based on Eclipse plug-in architecture.

But EasyPAB is still under development and desiderates more supplements and improvements, such as plug-ins for supporting newly emerged parallel platforms, supporting for more design patterns and skeletons, functions for debugging and performance analysis, etc. In the future work, we will focus on the visual modeling language and related translation technologies, to make EasyPAB be ready for practical use.

Acknowledgement. This work was supported by *Key Technologies R&D Program of Tianjin, China* (No. 06YFGZGX06000).

References

[AG1] Akon, M.M., Goswami, D., Li, H.F.: A model for designing and implementing parallel applications using extensible architectural skeletons. In: Malyshkin, V. (ed.) PaCT 2005. LNCS, vol. 3606, pp. 367–380. Springer, Heidelberg (2005)

[CM1] Chapman, B., Merlin, J., Pritchard, D., Bodin, F., Mevel, Y., Sorevik, T., Hill, L.: Program development tools for clusters of shared memory multiprocessors. Journal of Supercomputing 17(3), 311–322 (November 2000)

[CG1] Coxi, P.T., Glaser, H., Maclean, S.: A visual development environment for parallel applications. In: Visual Languages, Proceedings. 1998 IEEE Symposium, September 1–4, 1998, pp. 144–151 (1998)

[DV1] Bader, D.A., Kanade, V., Madduri, K.: SWARM: A Parallel Programming Framework for Multicore Processors. In: MTAAP 2007. Workshop on Multithreaded Architectures and Applications, Long Beach, CA, March 26-30, 2007, pp. 26–30 (2007)

[DA1] Goswami, D., Singh, A., Preiss, B.R.: From Design Patterns to Parallel Architectural Skeletons. Journal of Parallel and Distributed Computing 62(4), 669–695 (2002)

[Ge1] Geer, D.: Eclipse becomes the dominant Java IDE. Computer 38(7), 16–18 (2005)

[HJ1] Hawick, K.A., James, H.A.: A Java-based parallel programming support environment. In: Williams, R., Afsarmanesh, H., Bubak, M., Hertzberger, B. (eds.) High-Performance Computing and Networking. LNCS, vol. 1823, pp. 363–372. Springer, Heidelberg (2000)

[KT1] Koike, H., Takada, T., Masui, T.: VisuaLinda: a framework for visualizing parallel Linda programs. In: Visual Languages, Proceedings. 1997 IEEE Symposium, September 23-26, 1997, pp. 174–178 (1997)

[KR1] Asanovic, K., Bodik, R., Catanzaro, B., Gebis, J., Husbands, P., Keutzer, K., Patterson, D., Plishker, W., Shalf, J., Williams, S., Yelick, K.: The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report No. UCB/EECS-, -183.Electrical Engineering and Computer Sciences University of California at Berkeley, 2006.12 (2006)

[SM1] Siu, S., De Simone, M., Goswami, D., Singh, A.: Design patterns for parallel programming. In: PDPTA 1996. Proceedings of the 1996 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 230–240 (1996)

[SK1] Stankovic, N., Zhang, K.: A distributed parallel programming framework. Software Engineering, IEEE Transactions 28(5), 478–493 (2002)

[WL1] Wei, Z., Li, H.F, Goswami, D.: Composable skeletons for parallel programming. In: PDPTA 2004. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 1256–1261 (2004)

The Implementation of Parallel Genetic Algorithm Based on MATLAB*

Chen Guifen[1,2], Wan Baocheng[2], and Yu Helong[1,2]

[1] Institute of computer science and technology in Jilin university, 130118 Changchun, China
[2] Institute of information technology in Jilin agriculture university, 130118 Changchun, China

Abstract. This paper introduces the usage of MATLAB Distributed Computing Engine(MDCE). The relationship between the volume of the data transmitted and the transmission time is tested and the analysis of the data shows that there is a significant linear relationship between the two. Then we give an implemen-tation plan of the parallel genetic algorithm (PGA), and we also carried on the computation of a TSP example which shows a higher speedup and a better per-formance. All these show that the it is efficient and effective to use MATLAB to develop distributed computing application program.

Keywords: distributed computing, genetic algorithm, MATLAB.

1 Introduction

MATLAB is an outstanding scientific and engineering computing software owned by MathWorks Inc. It becomes the standouts in computing software fields because its high-speed calculation, reliable, rich features and convenience programming. With the popularity of computer networks, most computers are connected into a network, and distributed computing has entered the times of pc(personal computer). Developing distributed and parallel applications based on MATLAB would take full advantage of the rich functions of Matlab and greatly reduce the difficulty and cost of development, and highly improve efficiency[1]. Mathworks Corporation has timely promoted of the distributed computing engine and toolbox[2, 3], moreover Matlab is a cross-platform product, and therefore Matlab actually constitute the distributed computing engine and toolbox in heterogenous environment. Message Passing interface (MPI)[4], the most important parallel program tool at present, has already become the industry standard in parallel programming. Matlab has also provided the basic support for MPI functions. The distributed computing engine and toolbox, the commercial product pro-moted by the MathWorks Inc, has some unmatched advantages compared with other matlab development environment[1, 5].

* Supported by National "863"project "agriculture knowledge grid" (No. 2006AA10Z245) and National "863"project "research and application of maize farming system"(No. 2006AA10A309).

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 676–683, 2007.

As an important member in the Evolutionary Algorithms (EA) family and meta-heuristic optimization method, GA shows good performance in a great deal of com-plex optimization problems[6, 7]. Solving some practical issues will slow down the process of the evolution of Sequential GA due to more number of individuals and substantial calculation in need, this will make it is hard to meet the practical require-ment. Therefore parallel the GA becomes a study hotspot in GA[8, 9]. Though Matlab provids a GA toolbox[10], it does not realize the parallel GA.

Aimed at this study hotspot, we easily develop the parallel GA make use of two toolbox together. The rest of the paper is organized as follows. Section 2 introduces some contents on MDCE. Section 3 gives a detailed description of the implementation process of the PGA followed by a practical example to test the validity of the algo-rithm. Finally, a brief summary is made.

2 Matlab Distributed Computing Engine

Before carrying out the distributed computing, we need to configure the computing environment. The basic process is: first, start the MDCE service in each computer involved, and then start Job Manager, start Worker leech on to one of the Job Manag-ers (Matlab Session involved in computing in the background will be started at the meanwhile). All the Workers leeched onto one of the Job Manage constitute a Matlab distributed computing environment, and several such environments can be constructed. Having started the Matlab in some computer, created a Job Manager example, we could carry out distributed and parallel computing take advantage of them. The basic architecture is as showed in Fig.1.

The workflow is as follow: Discomposing the task into several and then submitted them to Job Manager, then Job Manager will appropriately distribute them for evalua-tion to workers according to the number of the workers and how many workers are available. After workers complete the tasks, results will be

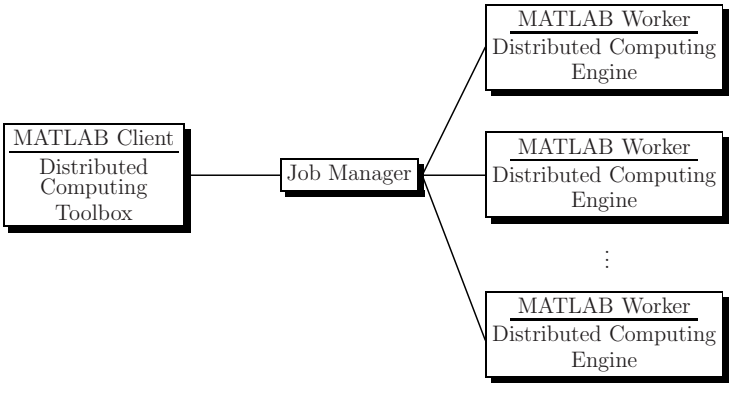


Fig. 1. Architecture of MDCE

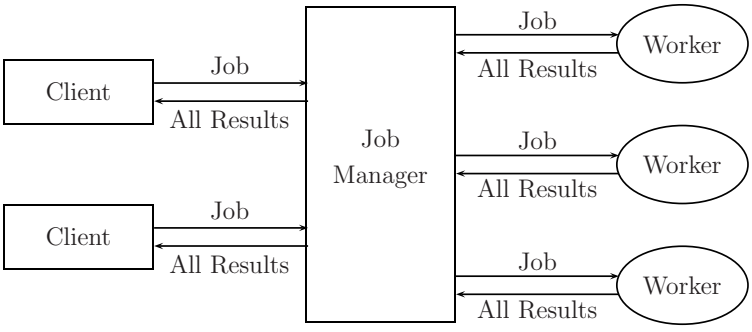


Fig. 2. AWorkflow of Distributed Computing

return to the Job Manager. After all the workers complete the tasks distributed to them, the job manager returns the results of all the tasks in the job to the client session. As is showed in figure 2.

Next, we will configure the Matlab distributed computing environment with two computers as an example. Similarly we can extend to more computers. We choose Windows XP SP2, node1 and node2 as our host computers' names. MDCE and tool-box are both version 3.1. Create a Job Manager on node1, then create a Worker in both nodes. Do as follow: click the start button-¿run, type the command cmd, a DOS window will appear, then input the following commands:

(1) Install MDCE
mdce install
(2) Start MDCE
mdce start
(3) Create Job Manager
start jobmanager-name jm-romotehost node1
(4) Create Worker
startjobmanager -name w1 -jobmanager jm -jobmanagerhost node1 -romotehost node1
startjobmanager -name w2 -jobmanager jm -jobmanagerhost node1 -remotehost node2

Now, start the Windows Task Manager at the nodes, you will see both processes mdced.exe and matlab.exe in the process option card. Note that close the firewall in the process of configuration and running MDCE.

3 Realization of the Parallel GA

At present, there are three main types of parallel GAs : global single-population mas-terslave GAssingle-population fine-grained and multiple-population coarse-grained GAs. In this paper, we give a realization of a coarse-grained parallel GA based on Matlab. The description of the algorithm is as follow.

```
Algorithm PGA:
Randomly generate a population P of size PopSz
Divide the population into num subpopulations of size sPopSz
Send each subpopulation and the parameter values to all nodes
for K cycles
   dopar(on nodes)
      for generation=1 to G do
         Run sequentially GA on subpopulation;
      endfor
      Send some individuals to the neighboring node
   endpar
endfor(for cycles)
```

Parameters involved above are explained as follow.
PopSz: size of the population.
sPopSz: size of the subpopulation.
num: number of nodes. K: times for the iteration of the population.
G: times for the iteration of the subpopulation.

In order for convenient use, the paralleled GA is realized in the form of function, which is similar with the genetic algorithm function provided by Matlab. The PGA function is as follow:

```
function [x, fval, population] = pag(jm, FUN, GenomeLength, Aineq,
    Bineq, Aeq, Beq, LB, UB, nonlcon, options)
```

Now we just simply explain the difference between the PGA function and the function ga. The first input parameter jm is a jobmanager object, input the following command in the Matlab command window.

```
jm = findResource('scheduler', 'configuration', 'jobmanager',
    'name', 'jm').
```

Parameter options could be attained from the function gaoptimset . The field related to stopping criteria will be discard, and the others will retain the same as before. Bute three fields(K,G and sPopSz)will be added.

This function mainly creates a parallel task object and sets some parameters related. The key is the task pgas established on it, which is the parallel executed m function file placed on the workers of each node. Now submit the file to the parallel job object and run it. Then fetch all the results ,which are made some appropriate process and returned to the output parameters. Finally, the parallel job object was destroyed.

Note: To make the function pgas run on both nodes, the FileDependencies field of the parallel job object should be set first,and then temporarily send the pgas to both nodes. Of course, copy the file pgas.m to both nodes advance will do, too. The form of the function is as follow,

```
function [x, fval, population] = pgas(FUN, GenomeLength, Aineq,
    Bineq, Aeq, Beq, LB, UB, nonlcon, options).
```

The meaning of the input and output parameters is the same as the function pga. This function is the core of the PGA , whose body is designed and developed by the PGA algorithm mentioned above and the MPI function provided by MATLAB. It mainly make use of the two MPI functions labSend and labRecieve. We could use the command :

```
[x fval exitflag output popu] = ga(opt.FUN, opt.GenomeLength, Aineq,
    Bineq, Aeq, Beq, LB, UB, nonlcon, opt)
```

to make genetic operation on each node. With circle structure, migration for the evolved subpopulation is made. Note that the sending command is standard form, which means that block occur or not depends on the state of the system. However, the receiving command is block form.

When a ring migration happens in the subpopulation, the number of the nodes should be even so that the blocking time can be reduced. In order to achieve that, the data should be transmitted as follow: as an example of six nodes, the data will be transmitted first between nodes linked by solid lines, then between nodes linked by dotted lines. No matter how many nodes there are, the total transmission time is nearly 2 times of single transmission time .As is showed in figure 3.

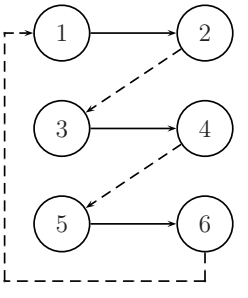


Fig. 3. AWorkflow of Distributed Computing

4 Testing Examples and Performance Analysing

Testing environment: CPU is PentinmIV 2.6GHz, the memory on the nodes that run jm is 1G, and the others are 512M.

First we test the relationship between transmission data volume and transmission time carried out by the MPI function labReceive provided by MATLAB. Transmis-sion time will be averaged between ten times. Data is showed as follow:

The model is $f(x) = p_1x+p_2$ using linear regression, x represents the volume of data transmitted, while $f(x)$ represents transmission time. The results returned by MATLAB are showed in table 2.

The result indicates that the data transmitted by MATLAB obey a linear relation-ship. The increase 1 kb, about 0.1 millisecond will be cost. The linear

Table 1. Transmission data volume and transmission time

transmission data volume(kb)	time(ms)	transmission data volume(kb)	time(ms)
703	65	1653	151
800	74	1800	166
903	83	1953	178
1013	93	2113	193
1128	105	2278	208
1250	115	2450	223
1378	125	2628	240
1513	139	2813	257

Table 2. Results of regression model

coef	value	with 95% confidence bounds	R-square
p_1	0.09075	(0.09018, 0.09133)	0.9999
p_2	1.329	(0.3123, 2.345)	

relationship supplies a basis for the analysis of the performance of the MATLAB parallel program.

Take TSP Att48[12] as an example, distance of the shortest path is 33524 in this problem. PGA is runing on the cluster constituted by one computer, two com-puters,four computers and eight computers respectively.

Theoretical calculation results and the test results are showed in table 3. Parameters in the sequential GA use the defaulted value of GATOOl. Select ring migration. The migration individuals, whose number is half of the number of the subpopulation, are randomly selected.

In order to test the state of the executive time of each section of the algorithm, some points in pga are chosen. Start the stopping clock on each nodes where the implemen-tation of pga begins, point before the migration of the population or after the evolution of the subpopulation, point after evolution of the population or before evolution of the subpopulation, point where the PGA ended on each node. The operation result statistic is showed in table 4. The total volume of transmission data is $25 \times 2 \times 48 \times 81024 = 937.5$ kb. The transmission time is 86 milliseconds according to the formula men-tioned above. It takes about 50

Table 3. PGA parameters

Number of nodes	2	4	8
K	25	25	25
G	40	20	10
Size of each subpopulation	100	100	100

Table 4. test results of PGA

Number of node	1	2	4	8
Optimum value	35061	34842	33869	33621
Total time(s)	9.4811	7.887	5.7582	4.6218
Pgas time(s)	/	5.9857	3.6845	2.3816
Start time(s)	/	1.0614	1.032	1.019
Total time of sequential ga in subpopultation	/	4.5877	2.3629	1.1037
Total of time(with block waiting time)(s)	/	0.13486	0.17882	0.19734

Table 5. speedup(without starting time)

Number of nodes	speedup	speedup(without start time)
1	1	1
2	1.6	1.9
4	2.6	3.6
8	4.0	7.1

percent of the total transmission time(with block wait-ing time). However, the block waiting time increases as the nodes increase, but the increasing trend is slow.

The total time consumed is about 2 seconds less than the pga consumed, as has nothing to do with the factors such as the number of nodes. It may be considered that the starting time of the whole parallel job is constant. Not including starting time, the pga speedup is showed in table 5.

5 Results and Discussion

(1) From the passage above we can see that developing distributed and parallel algorithm based on MATLAB platform will greatly reduce amount of the code and certainly increase the reliability of the code for a great deal of toolbox functions provided by MATLAB and what the users need to do is mainly to stitch these function or make a little modify. So the developing difficulty is decreased and the developing efficiency is increased. For scientific and engineering computing applications users, it is very favorable.
(2) For relatively fewer nodes, the practical result indicates that developing parallel program based on MDCE of MATLAB platform is more efficient.
(3) Grid Computing , considered to be the 3rd generation Internet technology, is trend of development of network technology in the future. The integration of the MATLAB and Grid Computing is a developing direction of the MATLAB distributed computing.
(4) Experiments are not carry out with relatively more nodes, where situation may change a little. Multi-nodes case is the future research direction.

References

1. Kepner, J., Ahalt, S.: MatlabMPI. Journal of Parallel and Distributed Computing 64, 997–1005 (2004)
2. The Mathworks, Inc.: MDCE3.1 System Administrator's Guide
3. The Mathworks, Inc.: Distributed Computing Toolbox 3.1 User's Guide
4. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard (November 15, 2003) http://www.mpi-forum.org/docs
5. Hoffbeck, J.P., Sarwar, M., Rix, E.J.: Interfacing MATLAB with a parallel virtual processor for matrix algorithms. The Journal of Systems and Software 56, 77–80 (2001)
6. Michalewicz, Z.: Evolution Programs——Genetic Algorithms + Data Structures. Science Press, Beijing (2000)
7. Chen, G., Wang, X.: Genetic Algorithm and Application. Posts & Telecom Press, Beijing (1996)
8. Prahlada Rao, B.B., Hansdah, R.C.: Extended Distributed Genetic Algorithm for Channel Routing. In: IEEE Trans on Neural Networks, pp. 726-733 (1933)
9. Hea, K., Zhengb, L., Donga, S., Tangc, L., Wud, J., Zheng, C.: PGO: A parallel computing platform for global optimization based on genetic algorithm. Computers & Geosciences 33, 357–366 (2007)
10. The Mathworks, Inc.: Genetic Algorithm Toolbox 2.1 User's Guide
11. Cantú-Paz, E.: A Survey of Parallel Genetic Algorithms. Technical Report. Department of Computer Science and Illinois Genetic Algorithms Laboratory,University of Illinois at Urbana-Champaign (1997)
12. Jiao, L., Du, H.: Immune Optimization, Sciense Press (2006)
13. Lim, D., Ong, Y.-S., Jin, Y., Sendhoff, B., Lee, B.-S.: Efficient Hierarchical Parallel Genetic Algorithms using Grid computing. Future Generation Computer Systems[J] 23, 658–670 (2007)

Composing Software Evolution Process Component*

Fei Dai and Tong Li

School of Information Science and Engineering, Yunnan University, Kunming 650091, China
flydai.cn@gmail.com, tli@ynu.edu.cn

Abstract. Composing software evolution process components into a complete software evolution process can effectively improve quality and efficiency of the software evolution process. However, existing researches do not propose a systematic method for composing software evolution process components. We propose a software evolution process component model (EPCM) which is based on 3C model and the concept of a software evolution process component (EPC). Based on EPCs, we propose three types of software evolution process component composition operations, namely, sequence composition, selection composition and concurrence composition.

Keywords: Petri Net, Component Model, Software Evolution Process, Component Composition, Process Reuse.

1 Introduction

As more and more successful software systems become legacy systems, software evolution becomes more and more important. On the one hand, software evolution has become an important characteristic in the software life cycle. On the other hand, software process plays an important role to increase efficiency and quality of software evolution. The term software evolution process denotes a set of interrelated software processes under which the corresponding software is evolving. A software evolution process provides a framework for managing activities that can very easily get out of control in software evolution, so we use software evolution processes to improve the effectiveness and efficiency of software evolution. Li [1] defined a formal evolution process meta-model (EPMM) based on extended Petri Net which is added with object-oriented technology and Hoare Logic to construct software evolution process models with four-level architecture. However, as more and more software evolution process modes are constructed by process designers based on EPMM [1], how we can reuse these existing software evolution models poses a challenging and exciting question for us.

* The project is supported by National Natural Science Foundation of China under Grant No. 60463002 and Education Science Foundation of Yunnan Province, China under Grant No. 04Z290D.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 684–692, 2007.

The term software evolution process reuse can be described as "usage of one process description in the creation of another process description" [10]. Osterweil presented a widely accepted view that software processes are software too [2]. According to Osterweil's idea, a software evolution process can be made up of many serial or parallel software evolution process components. Thus we apply component technology to software evolution processes and propose the concept of a software evolution process component (EPC). An EPC is actually an internally high cohesive and consistent software evolution process that can be reused with other EPCs to assemble a more powerful EPC. In order to describe an EPC, a software evolution process component model (EPCM) based on 3C model is proposed. Comparing to traditional software reuse, four essential steps for software evolution process reuse based on EPCs are needed. Firstly, we need to describe an EPC. Secondly, we need to search EPCs from software evolution process component library (EPCL) according to process requirements. Thirdly, we need a mechanism to tailor EPCs. Fourthly, we need a mechanism to compose EPCs into a software evolution process. In this paper, composing software evolution process component is focused on.

This paper is organized as follows. In the next section, a software evolution process models is proposed. In section 3, we propose three types of EPC composition operations, namely sequence composition, selection composition and concurrence composition. Finally, we conclude in section 4 with a brief summary and discussion of the future work.

2 Evolution Process Component Model

EPCM is the foundation of EPCL and is the key factor in realizing software evolution process reuse. Recently, the component models can be classified into three different categories according to their usage: (1) Model for component description/classification, such as REBOOT model [5]; (2) Model for component specification/composition, such as 3C model [4] and JBCOM [6]; (3) Model for component implementation, such as COM/DCOM [7][8], CORBA/OM [3], and Enterprise JavaBeans [9].

Obviously, it is difficult for us to build a comprehensive model to meet all software evolution processes defined by other process description languages. Thus we propose a component model to only meet the need of evolution process description language (EPDL) [1]. 3C model is a prescriptive component model that was proposed by Will Tracz on the "Reuse in Practice Workshop" in 1989. In 3C model, a component consists of three parts: concept, content, and context [4]. The concept is the abstract description of what a component does. The content describes how a component implements the concept. The context depicts the dependencies between the component and its environment. Based on 3C model, we propose EPCM which is shown in Figure 1.

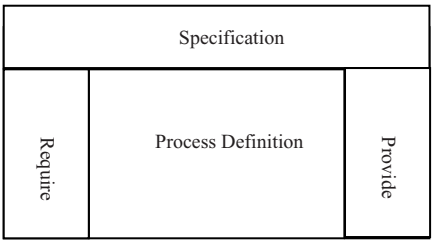


Fig. 1. Evolution Process Component Model

EPCM is a formal evolution process component model. The definition of EPCM is as follows:

Definition 1 EPCM is a 4-tuple epcm= (Req, Pro, Spec, PD)

1. Req and Pro are called the interfaces of EPC. Req denotes the required functions of EPC; Pro denotes the provided functions of EPC. They correspond to the concept in 3C model.
2. PD (Process Definition) is called the body of EPC, which is defined by EPDL [1]. It corresponds to the content in 3C model.
3. Specification is called the specification of EPC, which is used to describe the EPC briefly; it corresponds to the context in 3C model.

According to EPCM, the definition of EPC is as follows:

Definition 2 A EPC is a 7-tuple epc = (C, A; F, M_0, a_e, a_x , S,)

1. (C, A; F) is a net without isolated elements, $A \bigcup C \neq \Phi$;
2. C is a finite set of conditions; $\forall$ c$\in$ C is called a condition;
3. A is a finite set of activities; $\forall$ a$\in$ A is called an activity;
4. p=(C, A; F, M_0), called the body of epc, is a software evolution process with $M_0 = \Phi$;
5. a_e, $a_x \in$ A are called the entrance and the exit of EPC respectively, if $\exists$ step sequence $G_1G_2\ldots G_{n-1}$ (G_1, G_2, …, $G_{n-1} \subseteq$ A) and $\exists$ cases M_1, M_2, …, $M_n \subseteq$ C, such that $[a_e > M_1$, $M_1[G_1 > M_2$, …, $M_{n-1}[G_{n-1} > M_n$, $M_n[a_x >$ and $(M_n - a_x) = \Phi$;
6. S, called the mini specification, is a set of strings which is used to describe the epc briefly;

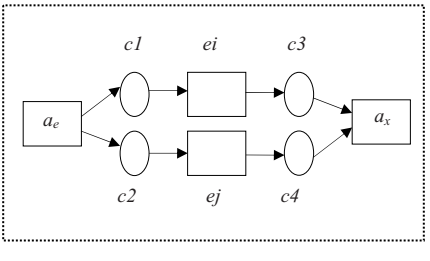


Fig. 2. An Evolution Process Component

From the definition above, we see that p corresponds to the PD in EPCM; a_e.I corresponds to the Req in EPCM; a_x.O corresponds to the Pro in EPCM; S corresponds to the Spec in EPCM. Here a_e.I denotes the input data structure of a_e, a_x.O denotes the output data structure of a_x [1]. Graphically, we represent activity as rectangle and condition as circle respectively. An EPC is shown in Figure 2.

The definition of EPC shown in Figure 2 is as follows:

epc = (C, A; F, M_0, a_e, a_x , S);

C = (c1, c2, c3, c4);

A = (ei, ej);

F=((a_e,c1), (a_e,c2), (c1,ei), (c2,ej), (ei,c3), (ej,c4), (c3, a_x), (c4, a_x));

M_0 = Φ ;

a_e= a_e;

a_x = a_x;

S=(…..);

The description of EPC shown in Figure 2 is defined by EPDL [1] as follows:

PROCESS An Evolution Process Component

Begin

ENTRANCE { a_e }

EXIT { a_x }

MINI SPECIFICATION

S={....};

CONDITION SET

C:={c1, c2, c3, c4};

ACTIVITY SET

A:={ a_e, ei, ej, a_x };

ARC SET

F:={(a_e,c1), (a_e,c2), (c1,ei), (c2,ej), (ei,c3), (ej,c4), (c3, a_x), (c4, a_x)};

MARKING { Φ } ;

END;

3 A Systematic Method for EPC Composition

EPC composition is defined as composing EPCs into a complete software evolution process. After process designers find out the required EPCs from EPCL, EPC composition is the next step. In this paper, we define three types of EPC composition operations, namely sequence composition, selection composition, and concurrence composition. In the following, we will compose R and S into T using these three composition operations. R and C are EPCs as shown in Figure 3.

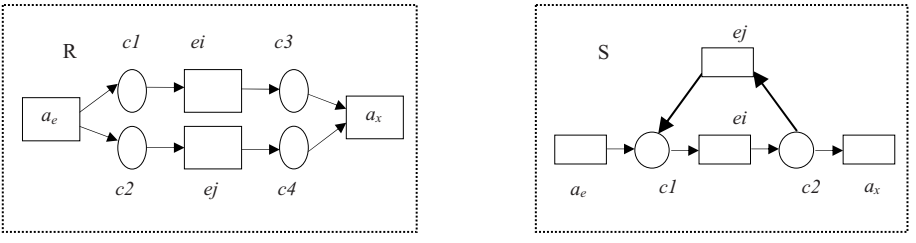


Fig. 3. Evolution Process Component R and C

3.1 Sequence Composition

Sequence composition is defined as composing R and S into T and T's execution sequence is that after R terminates, S then executes. During sequence composition, process designers should avoid interface conflict. The term interface conflict means that the interfaces between EPCs are mismatched. The following conditions are used for interface checking. By interface checking, process designers can check whether all the interfaces among EPCs are matched. If EPCs satisfy the following conditions, they are considered to be interface conformance. If EPCs are interface conformance, they can be composed into a more powerful EPC. The following algorithm 1 is used for sequence composition and Figure 4 shows the process of sequence composition.

The conditions are as follows:

-$R.a_e = S.a_x$;
-$R.a_e$ is a part of $S.a_x$

```
Algorithm 1 Fun SequenceComposing(R, S)
//Supposing that R.ax and S.ae are interface conformance
Begin
 // New(c) denotes the added conditions.
T.C = R.C + S.C + New(C);
T.A = R.A + S.A;
// New(F) denotes the added flow relations.
```

```
T.F = R.F + S.F + New(F);
T.M0 = R.M0 + S.M0;
T.ae = R.ae;
T.ax = S.ax;
T.S =R.S ∪ S.S;
return T;
End;
```

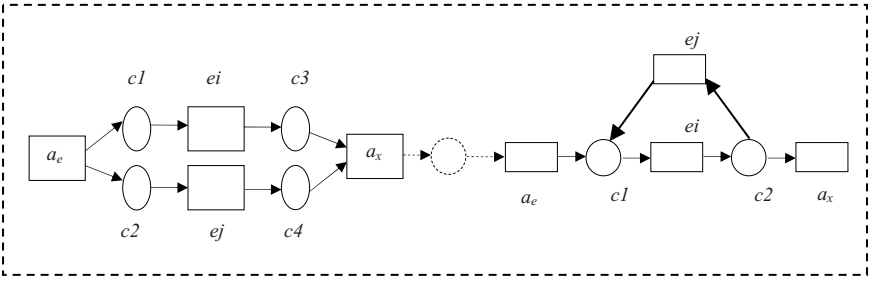


Fig. 4. Sequence Composition

3.2 Selection Composition

Selection composition is defined as composing R and S into T and T's execution sequence is that only R or S can execute according to process requirements. The following algorithm 2 is used for selection composition and Figure 5shows the process of selection composition.

```
Algorithm 2 Fun SelectionComposing(R, S)
Begin
T.C = R.C + S.C + c5 + c6;
T.A = R.A + S.A + a1 + a2;
// New(F) denotes the added flow relations.
T.F = R.F + S.F + New(F);
T.M0 = R.M0 + S.M0;
T.ae = a1;
T.ax = a2;
```

```
a1.I = R.a_e.I + S.a_e.I;
a1.O= R.a_e.O + S.a_e.O;
a1.L= R.a_e.L + S.a_e.L;
a2.I= R.a_x.I + S.a_x.I;
a2.O= R.a_x.O + S.a_x.O;
a2.L= R.a_x.L + S.a_x.L;
T.S =R.S ∪ S.S;
return T ;
End;
```

Algorithm 2 will introduce new activities and new conditions when running selection composition. It is necessary to notice that the newly added activities are virtual activities and the newly added conditions are virtual conditions. These activities have no actual operations in T except for passing a token from a condition to another condition. These conditions have no other meanings in T except for connections.

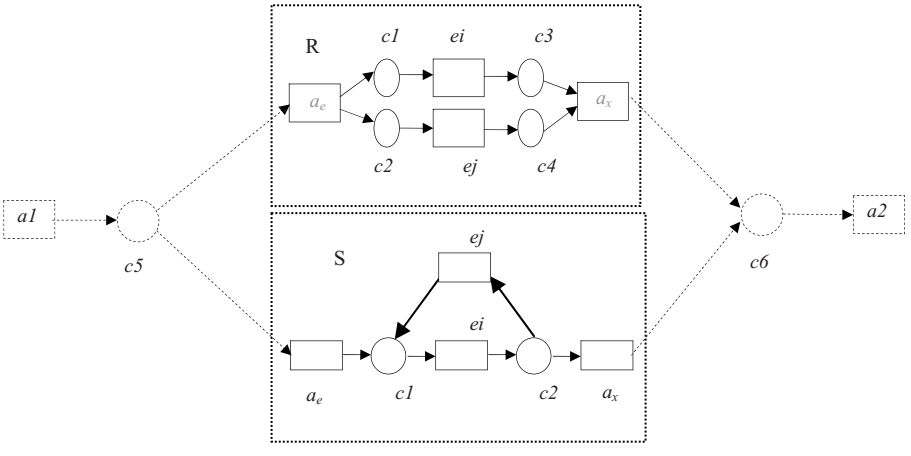


Fig. 5. Selection Composition

3.3 Concurrence Composition

Concurrence composition is defined as composing R and S into T and T's execution is that R and S can execute concurrently. The following algorithm 3 is used for concurrence composition and Figure 6 shows the process of concurrence composition.

```
Algorithm 3 Fun ConcurrenceComposing(R, S)
Begin
T.C = R.C + S.C + c5 + c6 + c7 + c8;
T.A = R.A + S.A + a1 + a2;
// New(F) denotes the added flow relations.
T.F = R.F + S.F + New(F);
T.M0 = R.M0+ S.M0;
T.ae = a1;
T.ax = a2;
a1.I = R.ae.I + S.ae.I;
a1.O= R.ae.O + S.ae.O;
a1.L= R.ae.L + S.ae.L;
a2.I= R.ax.I + S.ax.I;
a2.O= R.ax.O + S.ax.O;
a2.L= R.ax.L + S.ax.L;
T.S =R.S ∪ S.S;
return T;
End;
```

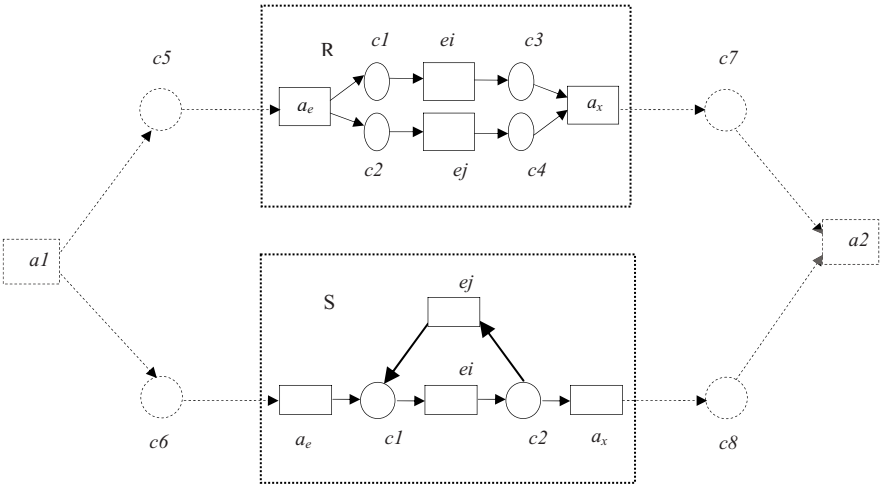


Fig. 6. Concurrence Composition

4 Conclusions

In this paper, the main idea is to compose EPCs into a complete software evolution process. In order to achieve the goal, we firstly propose an evolution process component model and define an evolution process component. Based on evolution process components, we propose three types of software evolution process component operations, namely, sequence composition, selection composition and concurrence composition.

However, there still remains much work. Firstly, a process operating system which is used to mange software evolution processes will be investigated. Secondly, after composing or tailoring EPCs, software evolution process's simulation will be researched.

References

1. Li, T.: Modeling formal software evolution process [Ph.D Thesis]. DeMontfort University (2007)
2. Osterweil, L.J.: Software Processes are Software too. In: Proceedings of the 9th International Conference on Software Engineering, pp. 2–13. ACM Press, New York (1987)
3. Object Management Group home page [online].Available WWW URL, http://www.omg.org
4. Implementation Working Group Summary, Reuse in Practice Workshop. Pittsburgh, Pensylvania (July 1989)
5. Weighted Term Spaces for Related Search. In: CIKM 1992. proceedings of the 1st International Conference on Information and Knowledge Management, pp.5–8 (November 1992)
6. JadeBird Project Group, JadeBird Component Model, Technical report, Department of Computer Science and Technology, Peking University (1997)
7. Microsoft Corporation. The Component Object Model Specification, Version 0.9, (October 24, 1995) Available WWW URL: http://www.microsoft.com/oledev/
8. Microsoft Corporation. Distributed Component Object Model Protocol COM/1.0, draft (November 1996) Available WWW. URL: http://www.microsoft.com/oledev/
9. Sun Microsystems, Inc., Enterprise JavaBeans Specifications Version 1.1, Available WWW. URL: http://java.sun.com/products/ejb/docs.html
10. Hollenbach, C., Frakes, W.: Software Process Reuse in an Industrial Setting, Fourth International Conference on Software Reuse, Orlando, FL, IEEE Computer Society Press, Los Alamitos, CA, pp. 22-30 (1996)

Asynchronous Spiking Neural P System with Promoters

Zhimin Yuan and Zhiguo Zhang*

Department of Computer Science
Sun Yatsen University
Guangzhou City, 510275 China
lnszzg@mail.sysu.edu.cn

Abstract. A new class of parallel computing devices called asynchronous spiking neural P systems with promoters is introduced. Unlike the regular spiking neural P systems, they work without help of the universal clock but they include a set of promoters. The computing power of these systems is proved to be Turing complete when they are considered as number generators. When they are considred as language generators, it is proved that any regular language can be a coding morphic image of the generated language of an asynchronous spiking neural P system.

Keywords: Molecular Computing, Membrane Computing, P System, Parallel Computing, Asynchronous P System, Turing Complete.

1 Introduction

Membrane systems (currently called P systems) are parallel computing devices inspired by the structure and the functioning of living cells (see [9]). Recently, a new class of membrane systems called spiking neural P system (SN P system, for short), with motivations related to the way neurons communicate by means of spikes, was introduced in [2]. An SN P system consists of a set of neurons placed in the nodes of a graph and sending signals (spikes) along synapses (edges of the graph), under the control of firing rules. One also uses forgetting rules, which remove spikes from neurons. One neuron is designated as the output neuron of the system and its spikes can exit into the environment. SN P systems can behave both as number generators and language generators.

The SN P systems as number generators was investigated in several papers (see, e.g., [2, 8]). Most of them considered the intervals between consecutive spikes (with several alternatives) as the numbers computed by the systems and all are proved to be Turing complete. The representations of finite, regular, and recursively enumerable languages were obtained in [4].

The SN P systems as language generators was also discussed in several references (see, e.g., [7, 8]). The generated language can be considered as spike trains (the sequence of moments when a neuron emits a spike) or trace languages (recording the

* Corresponding author, his work is partially supported by Guangdong Key Laboratory of Information Security.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 693–702, 2007.

labels of the neurons where the "marked" spike is present). [5] has shown that each regular language is the morphic image of a trace language intersected with a very particular regular language, while each recursively enumerable language over the one-letter alphabet is the projection of it. Similar conclusions were also obtained in [3], if we generalized the system by using extended rules (several spikes can be produced by a rule).

SN P systems are defined to work in synchronous model and universal clock plays an essential role for the result generated by the systems depends on it. In [12], a class of asynchronous SN P systems is mentioned. Global clock is remained but any neuron of the systems is assumed to be free to use a rule or not in each time unit. The systems are proved to be Turing complete by using extended rules.

In this paper, we introduce a class of asynchronous SN P systems with promoters (APSN P system, for short), a new parallel computing devices model which is more realistic and applicable. The universal clock is ignored but a set of promoters are introduced. The new system works in the following way: *each neuron gets fired (applies the rule) as soon as the required spike and promoter arrives, but spikes (sends produced spike and promoter to the related neuron) at any time*. We assume that no neuron can receive spikes and promoters from more than one neuron at a time and the neuron keeps closed if it contains the rule got fired but not spiked.

APSN P systems can behave as the number and language generators as SN P systems. Instead of considering the intervals between consecutive spikes, we assume the total number of spikes in the output neuron in the halting configuration is the number generated. The systems are proved to be Turing complete by using only restrict rules (rules only produce one spike) when acting as number computing devices and we will also show that each regular language is the morphic image of the trace language for certain APSN P system.

In the next section, we give the preliminary definitions used in the subsequent section of the paper. Section 3 introduces the asynchronous spiking neural P system with promoter. In Section 4, we prove that APSN P system are computationally complete, while in Section 5 we show that each regular language is the morphic image of the trace language for certain APSN P system. We end (Section 6) with a few comments and suggestions for further investigation.

2 Preliminary Definitions

We introduce here some notations used later in the paper. For an alphabet V, V^* denotes the set of all finite strings of symbols from V; the empty string is denoted by λ, and the set of all nonempty strings over V is denoted by V^+. When $V = \{a\}$ is a singleton, then we write simply a^* and a^+ instead of $\{a\}^*$, $\{a\}^+$.

A morphism h: $V_1^* \to V_1^*$ such that $h(a) \in \{a, \lambda\}$ for each $a \in V_1$ is called projection, and a morphism h: $V_1^* \to V_2^*$ such that $h(a) \in V_2 \cup \{\lambda\}$ for each $a \in V_1$ is called a weak coding; it is a coding if $h(a) \in V_2$ for all $a \in V_1$.

A Chomsky grammar is given in the form $G=(N,T,S,P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $S \in N$ is the axiom, and P is the

finite set of rules. For regular grammars, the rules are of the form $A \to aB, A \to a$, for some $A, B \in N, a \in T$.

We ignore here the empty string by convention. We denote by *FIN*, *REG*, *CF*, *CS*, *RE* the families of finite, regular, context-free, context-sensitive, and recursively enumerable languages respectively. The family of Turing computable sets of numbers is denoted by *NRE* (these sets are length sets of *RE* languages, hence the notation).

The universality result in this paper is based on the notion of simulating a register machine. Such a device – in the non-deterministic version – is a construct $M = (m, H, l_0, l_h, I)$, where m is the number of registers, H is the set of instruction labels, l_0 is the start label (labeling an ADD instruction), l_h is the halt label (assigned to instruction HALT), and I is the set of instructions; each label from H labels only one instruction from I, thus precisely identifying it. The instructions are of the following forms:

- l_i: *(ADD(r), l_j, l_k)* (add 1 to register r and then go to one of the instructions with labels l_j , l_k non-deterministically chosen),
- l_i: *(SUB(r), l_j, l_k)* (if register r is non-empty, then subtract 1 from it and go to the instruction with label l_j, otherwise go to the instruction with label l_k),
- l_h: *HALT* (the halt instruction).

A register machine M generates a set $N(M)$ of numbers in the following way: we start with all registers empty (i.e., storing the number zero), we apply the instruction with label l_0 and we continue to apply instructions as indicated by the labels (and made possible by the contents of registers); if we reach the halt instruction, then the number n present in register 1 at that time is said to be generated by M. (Without loss of generality we may assume that in the halting configuration all other registers are empty; also, we may assume that register 1 is never subject of SUB instructions, but only of ADD instructions.) It is known that register machines generate all sets of numbers which are Turing computable.

A spiking neural P system (abbreviated as SN P system), of degree $m \geq 1$, is a construct of the form

$$\Pi = (O, \sigma_1, ..., \sigma_m, syn, i_0),$$

where:

1. $O = \{a\}$ is the singleton alphabet (a is called spike);
2. $\sigma_1, ..., \sigma_m$ are neurons, of the form

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m,$$

where:

a) $n_i \geq 0$ is the initial number of spikes contained by the neuron;

b) R_i is a finite set of rules of the following two forms:

(1) $E / a^c \to a; d$, where E is a regular expression over $O, c \geq 1$, and $d \geq 0$;

(2) $a^s \to \lambda$, for some $s \geq 1$, with the restriction that $a^s \in L(E)$ for no rule of type (1) from R_i;

3. $syn \subseteq \{1, 2, ..., m\} \times \{1, 2, ..., m\}$ with $(i, j) \notin syn$ for $1 \leq i \leq m$ (synapses);
4. $i_0 \in \{1, 2, ..., m\}$ indicates the output neuron.

The rules of type (1) are firing (we also say spiking) rules, and they are applied as follows: if the neuron contains k spikes, $a^k \in L(E)$ and $k \geq c$, then the rule $E/a^c \rightarrow a;d$ can be applied, and this means that c spikes are consumed, only k-c remains in the neuron, the neuron is fired, and it produces a spike after d time units (a global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized). If d=0, then the spike is emitted immediately, if d =1, then the spike is emitted in the next step, and so on. In the case $d \geq 1$, if the rule is used in step t, then in steps t, $t + 1$, $t + 2$,…, $t + d - 1$ the neuron is closed, and it cannot receive new spikes (if a neuron has a synapse to a closed neuron and tries to send a spike along it, then the spike is lost). In step $t + d$, the neuron spikes and becomes open again, and hence can receive spikes (which can be used in step $t + d + 1$). A spike emitted by a neuron σ_j replicates and goes to all neurons σ_j such that $(i, j) \in syn$. If in a rule $E/a^c \rightarrow a;d$ we have $L(E) = \{a^c\}$, then we write it in the simpler form $a^c \rightarrow a;d$.

The rules of type (2) are forgetting rules, and they are applied as follows: if the neuron contains exactly s spikes, then the rule $a^s \rightarrow \lambda$ can be used, and this means that all s spikes are removed from the neuron.

In each time unit, in each neuron that can use a rule we have to use a rule, either a firing or a forgetting one. Because two firing rules $E_1/a^{c_1} \rightarrow a;d_1$ and $E_2/a^{c_2} \rightarrow a;d_2$ can have $L(E_1) \cap L(E_2) \neq \varnothing.$, it is possible that two or more rules can be applied in a neuron, and then one of them is chosen non-deterministically. Note however that we cannot interchange a firing rule with a forgetting rule, as all pairs of rules $E/a^c \rightarrow a;d$ and $a^s \rightarrow \lambda$ have disjoint domains, in the sense that $a^s \notin L(E)$.

The rules are used in the non-deterministic manner, in a maximally parallel way at the level of the system: in each step, all neurons which can use a rule, of any type, spiking or forgetting, have to evolve, using a rule.

3 Asynchronous Spiking Neural P System with Promoter

We pass now from the previous definition of a SN P system to a new model of parallel computing system in which global clock is removed and a set of promoters is introduced. The new system works in the following way: each neuron gets fired as soon as the required spike and promoter arrives, but spikes at any time. The promoter is produced by a spiking rule and along with the spike, it passes to all neurons connected by a synapse to the spiking neuron. The promoter will enable certain (spiking or forgetting) rules and after spiking the rule the promoter may stay in the neurons, leave from it or melt in it. The formal definition is as follows:

Definition 1. We consider a *asynchronous spiking neural P system with promoters* (in short, an APSN P system), of degree m≥1, in the form

$$\prod = (O, P, \sigma_1, ..., \sigma_m, syn, i_0)$$

where:

1. $O=\{a\}$is the singleton alphabet (a is called *spike*);
2. P is the set of signal-promoters, with the restriction that $a \notin P$;
3. $\sigma_1,...,\sigma_m$ are neurons, of the form

$$\sigma_i = (n_i, S_i, R_i, R_i^{'}), 1 \le i \le m,$$

where:

a) $n_i \ge 0$ is the *initial number of spikes* contained by the cell;

b) $S_i \subseteq P$, is the initial promoters contained by the cell;

c) R_i is a finite set of *rules without promoter* of the following two forms:

(1) $E / a^r \to aq$, where E is a regular expression over O, $r \ge 1$, $q = \lambda$ (no promoter is produced) or $q \in P$ (one promoter is produced);

(2) $a^s \to \lambda$, for some $s \ge 1$

d) $R_i^{'}$ is a finite set of rules with promoter of the following two forms:

(1) $E / a^r \to aq|_{p,tar}$, where E is a regular expression over O, $r \ge 1$, $q = \lambda$ or $q \in P$, $p \in P$, $tar \in \{here, go, melt\}$;

(2) $a^r \to \lambda|_{p,tar}$, where E is a regular expression over O, $r \ge 1$, $p \in P$ and $tar \in \{here, go, melt\}$;

4. $syn \subseteq \{1,2,...,m\} \times \{1,2,...,m\}$ with $(i, j) \notin syn$ for $1 \le i \le m$ (synapses among cells);
5. $i_0 \in \{1,2,...,m\}$ indicates the output neuron.

Since the definition of usual spiking neural P system has been given in the previous section. We only explain the difference between the two.

Promoter is introduced and rules are classified into two parts: with and without promoters. Promoters can be distinguished, but they are present in the set sense, i.e. we cannot have more than one copy of the same promoter in one neuron. The promoter only generated by firing rules, not forgetting rules. Once it is generated, it will go along with the spike into the neurons that have connections with the one where is produced. The most important function of a promoter is that it enables the "rules with promoters". For instance, although regular expression E covers the content of the neuron, the rule of the form $E / a^r \to aq|_{p,tar}$ cannot be applied if promoter p does not exist. When the enabled rule spikes, the promoter will stay in previous neutron (if *tar=here*), go to relative ones (if *tar=go*), or simply disappear (if *tar=melt*). Promoter does not make any sense to the "rules without promoters" which defined the same as the rules in usual SN P system except that they can produce one promoter. If one promoter enters the neuron in which no rule is required for it, it will stay there forever without any use.

The firing rule can produce a promoter along with a spike. We define the rules of the form $E / a^r \to aq$ and $E / a^r \to aq|_{p,tar}$ are firing rules, while $a^s \to \lambda$ and $a^r \to \lambda|_{p,tar}$ are forgetting rule. Any firing rule can produce one promoter, but it is not indispensable, i.e. the rules of the form $E / a^r \to a$ are also included here.

Another difference of the rule from SN P system is that we remove the restriction that forgetting rule cannot be interchanged with a spiking rule, i.e. one of the spiking and forgetting rules will be chosen to be applied randomly in some situations.

Universal clock has been removed is the biggest variation, which enables the device works in an asynchronous mode. In SN P system, getting fired and spiking are two important actions take place in a step. The neuron gets fired at the next clock time when it receives the right number of spike, and spikes after certain time interval that is indicated in the rule. The neuron is closed (it can neither receive further spikes nor fire again) between the time of getting fired and spiking. Since global clock is ignored in APSN P system, we assume that the neuron gets fired as soon as some rule in it is applicable, (i.e. the neuron contains the required number of spikes and certain kinds of promoter if necessary) but it spikes at any time. The neuron is closed during the time when it has got fired but not spiked. Another assumption is no neuron can receive spikes or promoters from more than one neuron at a time. If the spike and promoter (if any) one neuron has received enable the rule in it, the neuron will be closed immediately, thus the spikes and promoters from other neuron may be missing. If the spike and promoter do not enable any rule, then the neuron keeps open, waiting for other spikes and promoters to arrive.

4 APSN P System as Number Generator

An APSN P system can behave as a number generator. Initially, some spikes and promoters are contained in system. Then the system will run asynchronous until it reaches the halting configuration (all neurons are open, but no rule is applicable). The number of spikes in the output neuron is the number system has been generated. If the system can't reach the halting configuration, we define the computation failed.

We denote by $N_a(\Pi)$ the set of numbers computed by a APSN P system Π, and by $NAPSN_m^r(rule_k, cons_p, forg_q)$ the family for all sets $N_a(\Pi)$ computed as above by APSN P system with at most *r* promoters and *m* neurons, each neuron having at most *k* rules, each of the spiking rules consuming at most *p* spikes, and each forgetting rule removing at most *q* spikes. When one of the parameters *m*, *r*, *k*, *p*, and *q* is not bounded, then it is replaced with *

APSN P system will be proved to be powerful for it can generate all the numbers a Turning Machine can receive.

Theorem 1. $NRE \subseteq APSN_*^r(rule_k, cons_p, forg_q)$ for all $r \geq 5, k \geq 4, p \geq 1, q \geq 1$.

Proof. The proof of is based on constructing an SN P system Π that simulates a given register machine *M*. We construct two modules to simulate ADD and SUB instruction of register machine. These modules are presented in Figures 1, 2. The neurons labeled l_i are associated with the instruction label in *M* (and all these neurons contain the same rules). The neurons labeled r_i are associated with the registers in *M* and the number of the spikes in neuron r_i is exactly the value in register r_i in *M*. Other neurons labeled by c_i are also introduced for some special use.

Initially, (1) all neurons contain no object (but some contain certain promoters), with the single exception of the neuron l_0 (the label of the initial instruction of *M*),

which contains only one spikes (one copy of *a*) and (2)all neurons label with l_i hold some promoters: if l_i is correspond to ADD instruction, then neuron l_i contains promoter *d*; if l_i is correspond to SUB instruction, then neuron l_i contains promoter *b*; and if l_i is correspond to HALT instruction, then neuron l_i contains promoter *h*.

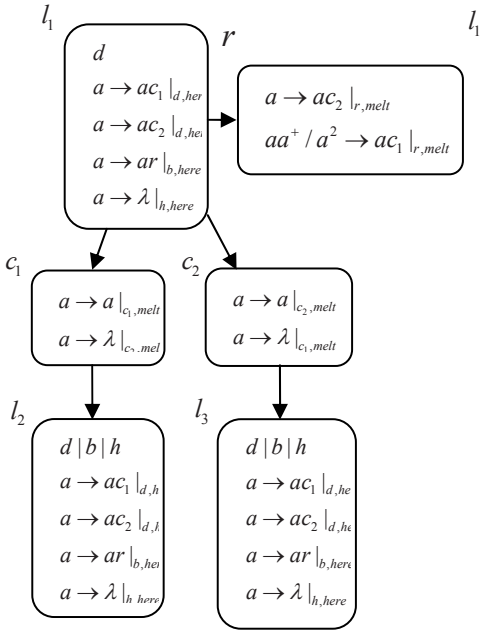


Fig. 1. Module ADD

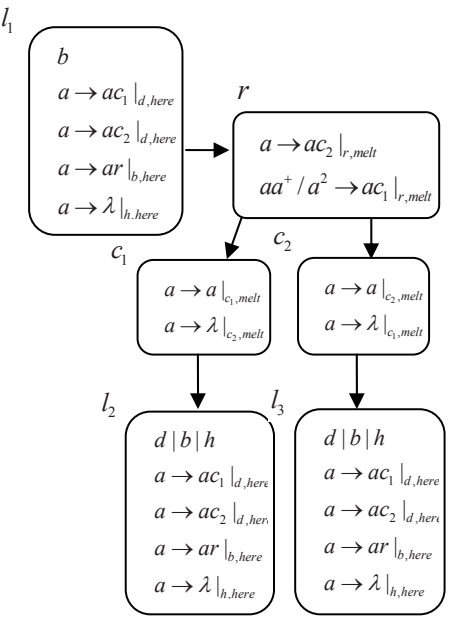


Fig. 2. Module SUB

Simulating an ADD instruction $l_i : (ADD(r), l_j, l_k)$

In neuron l_1, the only promoter exist is *d*. Thus, one of the rules $a \to ac_1 |_{d,here}$, $a \to ac_2 |_{d,here}$ will be applied indeterminately. If rule $a \to ac_1 |_{d,here}$ applied, the spike will go to l2, (for the spike pass through c1, while be forgotten in c2). If rule $a \to ac_1 |_{d,here}$ applied, the spike will go to l3 (for the spike pass through c1, while be forgotten in c2). Note that neuron ri receive promoter c1 or c2 as well, and it will stay there forever without any use.

Simulating an SUB instruction $l_i : (SUB(r), l_j, l_k)$

The only promoter exist in l_1 is *b*. Thus, $a \to ar |_{b,here}$ is the only rule that can be applied. After neuron l_1 spikes, a spike along with promoter *r* will enter neuron r_i, where either promoter c_1 or c_2 will be generated depending on the number of spikes in neuron r_i. If neuron r_i does not contain any spike, $a \to ac_2 |_{r,out}$ will be applied. After neuron r_i spikes, the number of the spikes in neural r_i does not change (remains zero), and neuron l_3 will get fired. If neuron r_i contains at least one spike,

$aa^+ / a^2 \to ac_1 |_{r,out}$ will be applied, the number of spike in it will decrease by 1 and neuron l_2 will get fired. The neuron labeled with c_1, c_2 have the some function as in ADD module.

Ending a computation

When neuron l_h (which contains promoter h) receive a spike. The rule $a \to \lambda |_{h,here}$ will be applicable, for no other promoter but h is available. After it applies, there will be no spike in the system and the computation halts. The number of spikes in neuron r_1 is exactly the value of register 1 of M. Consequently, $N_a(\Pi) = N(M)$ and this completes the proof. □

5 APSN P System as Language Generator

APSN P system generates language by following the traces of a distinguished spike in its journeys. We "mark" one spike in a neuron and follow its path during the computation, recording the labels of the neurons where this spike is present after the previous neuron has spiked. The string of labels of the neurons through which the marked spike has passed through is the language generated by the system when a computation successfully finished. The computation is successful only if the system reaches the halting configuration or the marked spike has disappeared.

Because global clock is omitted, we assume only one copy of the symbol associated with labeled neuron is generated no matter how long the marked spike stays there, which is different from the definition of trace language in usual SN P system. Since no neuron has the synapse to itself, the identical symbol cannot appear consecutively in the language generated in this way. The output neutron i_0 can be omitted here.

We denote by $T_a(\Pi)$ the language of all strings describing the traces of the marked spike in Π, and by $TAPSN_m^r(rule_k, cons_p, forg_q)$ the family for all sets $T_a(\Pi)$ computed as above by APSN P system with at most r promoters and m neurons, each neuron having at most k rules, each of the spiking rules consuming at most p spikes, and each forgetting rule removing at most q spikes. As usual, the parameters r, m, k, p, q is replaced with * if it is not bounded.

In the following, we will prove each regular language is the morphic image of the language generated by an APSN P system.

Theorem 2. For each regular language $L \subseteq V^*$, there is an APSN P system Π, such that $L \subset h(T(\Pi))$, for some coding h; actually, $T(\Pi) \in TAPSN_{m_L}^{r_L}(rule_{k_L}, cons_1, forg_1)$, for some constants r_L, m_L, k_L depending on language L.

Proof. Let us take a regular grammar $G = (N, V, A_1, P)$ generating the language L with $V = \{b_1, ..., b_n\}$. There exist another regular grammar such that $G' = (N', V', A_0, P')$. We constructed G' as follows: Firstly, let $N' = N$, $V' = V \cup b_0, P' = P$. If the set P'

contains a rule of form $A_p \to b_i A_p (A_p \in N)$, we delete it, and add a new nonterminal A' into N' and two rules of the form $A_p \to b_i A', A' \to b_0 A_p$ into P'. If set P' contains a couple of rules of form $A_p \to b_i A_q, A_q \to b_i A_r$, we delete the rule $A_q \to b_i A_r$, and add a new nonterminal A'' into N' and two rules of the form $A_q \to b_0 A'', A'' \to b_i A_r$ into P'. We did it continuously until no rules of the form above exist in P'. At last, we add a new start symbol A_0 into N' and a rule $A_0 \to b_0 A_1$ in to P'.

Thus, we have $G' = (N', V', A_0, P')$ with $N' = \{A_0, A_1 ... A_m\}$, $V = \{b_0, b_1, ..., b_n\}$, and $P = \{A_p \to b_i A_q, A_p \to b_i\}$ for $0 \le p, q \le m, 0 \le i \le n$. For $h: b_i \to b_i, b_0 \to \lambda$, we have $L(G)=h(L(G'))$. $L(G')$ cannot generate such language that $Xb_i b_i Y$, but $Xb_i b_0 b_i Y$ (X,Y are the strings in which no identical symbol appear consecutively). Next, we will prove there is an APSN P system Π that generates $L(G')$.

We labeled the neuron with nonterminal b_i and use terminal A_p as promoters. We construct the following Π simulating G'

$$\Pi = (O, P, \sigma_{b_0}, ..., \sigma_{b_n}, syn),$$

where:

1. $O=\{a\}$;
2. $P = N'$;
3. $\sigma_{b_i} = (0, \varnothing, \varnothing, R), 1 \le i \le n$, $\sigma_{b_0} = (1, A_0, \varnothing, R)$, where $R = \{a \to aA_q \mid_{A_p, melt}, a \to \lambda \mid_{A_p, melt}, a \to \lambda \mid_{A_i, melt}\}$ for $A_p \to b_i A_q, A_p \to b_i, l \ne p$;
4. $syn = \{< i, j > \mid A_p \to b_i A_q, A_q \to b_j$ or $A_q \to b_j A_r\}$ for $1 \le p, q, r \le m$ and $1 \le i, j \le n$.

The computation begins in neuron σ_{b_0}. The trace of the marked spike is the language generated by the system. If the "marked" spike disappeared because of forgetting rule instead of being missing during computation, the generated trace language belongs to $L(G')$. Otherwise, it does not belong to $L(G')$.

Therefore, $L(G') \subset T(\Pi)$, and we have $L(G) = h(L(G')) \subset h(T(\Pi))$ for $h: b_i \to b_i, b_0 \to \lambda$. □

6 Conclusion

We have introduced a class of parallel computing devices called asynchronous spiking neural P system with promoter: universal clock has been ignored and each neuron gets fired as soon as some rule in it is applicable, but it spikes at any time. We have proved that APSN P system is universal when it is considered as a number generator. We have also shown that each regular language is the morphic image of the language generated by an APSN P system if it behaves as a language generator.

Many topics remain to be investigated, starting with the problem whether or not asynchronous SN P system is Turing complete without using promoters. Another variant of interest is to see whether or not languages from other families (e.g. CF, CS, and RE) can be represented starting form languages generated by APSN P system. It

is also of interest to consider using other ways (instead of tracing the spike) to generate the language and discussing the power of such language.

References

[1] Cavaliere, M., Sburlan, D.: Time-independent P systems. In: Mauri, G., Păun, G., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (eds.) WMC 2004. LNCS, vol. 3365, pp. 239–258. Springer, Heidelberg (2005)
[2] Ionescu, M., Păun, Gh., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae 71(2-3), 279–308 (2006)
[3] Chen, H., Ishdorj, T.O., Păun, Gh., et al.: Spiking neural P systems with extended rules. In: Gutiérrez-Naranjo, M.A., et al. (eds.) Proceedings of Fourth Brainstorming Week on Membrane Computing, Sevilla, pp. 241–265 (2006)
[4] Chen, H., Freund, R., Ionescu, M., et al.: On string languages generated by spiking neural P systems. In: Gutiérrez-Naranjo, M.A., et al. (eds.) Proceedings of Fourth Brainstorming Week on Membrane Computing, Sevilla, vol. I, pp. 169–194 (2006)
[5] Chen, H., Ionescu, M., Păun, A., Păun, Gh., Popa, B.: On trace languages generated by spiking neural P systems. Proc. Fourth Brainstorming Week on Membrane Computing, Sevilla (2006)
[6] Păun, Gh.: Twentry Six Research Topics About Spiking Neural P Systems. In: Fifth Brainstorming Week on Membrane Computing, Sevilla, Spain (2007)
[7] Păun, Gh.: Languages in membrane computing. Some details for spiking neural P systems. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 20–35. Springer, Heidelberg (2006)
[8] Păun, Gh., Perez-Jimenez, M.J., Rozenberg, G.: Spike trains in spiking neural P systems. Intern. J. Found. Computer Sci. 17(4), 975–1002 (2006)
[9] Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences 61,1 (2000) 108-143, and TUCS Research Report 208 (1998)
[10] Sburlan, D.: Clock-free P systems. In: Pre-Proceedings of WMC5, Fifth Workshop on Membrane Computing, Milano, pp. 372-383 (2004)
[11] Cavaliere, M.: Toward asynchronous P systems. In: Pre-Proceedings of WMC5, Fifth Workshop on Membrane Computing, Milano, pp. 161-173 (2004)
[12] Cavaliere, M., Egecioglu, O., Ibarra, O.H., Woodworth, S., Ionescu, M.: Gh. Păun, Asynchronous Spiking Neural P Systems, Technical Report 9/2007, Microsoft Research - University of Trento, Centre for Computational and Systems Biology
[13] The P Systems Web Page, http://psystems.disco.unimib.it

Fingerprint Classification Method Based on Analysis of Singularities and Geometric Framework

Taizhe Tan, Yinwei Zhan, Lei Ding, and Sun Sheng

Faculty of Computer, Guangdong University of Technology, Guangzhou 510090, China
tantaizhe@263.net

Abstract. According to the former contributions, the authors present a novel fingerprint classification method based on analysis of singularities and geometric framework. First, a robust pseudoridges extraction algorithm on fingerprints is adopted to extract the global geometric shape of fingerprint ridges of pattern area. Then, by use of the detected singularities and with the help of the analysis of the global geometric shape of fingerprint ridges of pattern area, the fingerprint image is classified into different pre-specified classes. This algorithm has been tested on the NJU fingerprint database which contains 2500 images. For the 1000 images in this database, the classification accuracy is 92.2%.

Keywords: fingerprint classification, global geometric shape, singularity.

1 Introduction

Nowadays, automatic fingerprint identification is one of the most reliable and important biometric technology. However, automatic fingerprint identification is computationally demanding especially for a large database. So an effective fingerprint indexing scheme can greatly help facilitate efficient matching for large fingerprint database. Fingerprint classification, which classifies fingerprint images into a number of pre-defined categories, provides an indexing scheme to improve the efficiency of the matching task [1~2, 22].

Over the past few decades, a significant number of approaches have been developed for the purpose of fingerprint classification. The methods of fingerprint classification can fall into the following categories:(i) the knowledge-based approach[1, 3~4], (ii) the structural approach[5~9], (iii) the frequency-based approach[10],(iv) the syntactic approach[11~13], and (v) the artificial neural network approach[14~17]. Furthermore, the hybrid approach[1~2] that combines two or more of the foregoing approaches are combined, is used to accomplish the classification task.

Most of the approaches mentioned above are subjected to one kind of disadvantage. One apparent weakness of these approaches is that they are not able to deal with the usual noisy characteristic of fingerprint images well. For instance, due to the core and delta points used in classification might be uncertain in noisy images or be dropped in non-whole images, only using the singularities for fingerprint classification might not be viable. Many approaches may be susceptible to the large variations of ridge orientation within the patterns of the same class, or they would only work

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 703–712, 2007.

within a fixed state of orientation and position. In a fingerprint image, the local ridge features carry the individuality information about the fingerprints and the global pattern configurations, which form special patterns of ridges and furrows in the central region of the fingerprint, carry information about the fingerprint class. Therefore, the global pattern configurations should be used for fingerprint classification. Generally, global fingerprint features can be derived from the orientation field and global ridge geometric framework. The orientation field of a fingerprint consists of the ridge orientation tendency in local neighborhoods, and it is highly structured and can be roughly approximated by a core-delta model[2]. Therefore, singularities and their relationship can be used for fingerprint classification. On the other hand, the global ridge geometric framework also provides important clues as to the global pattern configuration of a fingerprint image [4].

This paper presents a fingerprint classification method based on the analysis of singularities and geometric framework. First, a robust pseudoridges extraction algorithm on fingerprints is adopted to extract the global geometric shape of fingerprint ridges of the pattern area. Then, by using the singularities and the global geometric shape features, the fingerprint image is classified into six classes (arch, tented arch, left loop, right loop, whorl and twin). Because it is difficult to get the correct and complete information of singularities used in the other classification method, this method is not merely based on the singularities and thus improves the classification accuracy. The global geometric shape of fingerprint ridges is extracted by directly tracing the orientation field, the traced orientation field is smoothed locally, and the global ridge geometric framework traced remains constant under large variations of local ridge orientation, Therefore, the fingerprint classification algorithm is robust. Moreover, our algorithm is invariant under transition and rotation.

In the following sections, the paper presents the details of our fingerprint classification method. In section 2, the global geometric shape extraction method is introduced. Section 3 presents our classification scheme. Experimental results and conclusions are given in section 4.

2 Global Geometric Shape Extraction

In our fingerprint classification scheme, the feature information such as the global ridge geometric framework, fingerprint singularities and the symmetrical axis of the core point will be used for fingerprint classification, but the methods for getting these features are not the emphasis of this paper, which have been reported in our other papers[18] [21]. The essential ideas for above several features extraction are given the brief introduction as follows:

2.1 Fingerprint Pseudoridge Tracing

According to the method presented in [18], the pseudoridge tracing is started from the core point and only uses the flow field information, avoiding complicated computation of thinned ridges, minutia and other trivial processing. Because the fingerprint is flow-modal, the flow field reflects the global tendency of fingerprint ridges and is continuous except in the individual singularity[19], as is different from the gray

ridges, some of which are conjoint, some disconnected and other furcated. Besides the ridge structures in poor quality fingerprint images are not robust. Hence, it is more reasonable to trace the flow field than to trace the ridges directly. In order to make the traced pseudoridges more accurate, the paper suggests a skillful and effective method for the orientation field estimate. On only the exact points traced but not the whole fingerprint image, orientation field estimates are performed, the operation time will be reduced as a result. The paper also exploits a method of adaptive tracing, in which if the variation of flow field is dull, the tracing step is large, otherwise (for example, near the core points), the tracing step is small. This characteristic is useful in checking the singularities. Moreover, our algorithm operation is invariant under transition and rotation for it is irrelative with a fixed state of orientation and position.

In paper [18], an effective point direction estimatation approach is also proposed. The approach utilizes the ridge orientations around an aim point to smooth the aim point direction skillfully, which reflects the general orientations of the neighbor ridges and alleviates the local noises accordingly. Thus, the proposed point direction estimatation approach ensures the pseudoridge tracing algorithm is robust to the noise images. The specific idea is as follows:

The point direction of a point is usually specified for a region (block) that centered at this point. In this paper, the 16×16 pixels region is used to compute the point directions. To get a more reasonable point direction of an aim point, a smoothing technology should be introduced to alleviate the effect of noise. The directions of the points, which are located in the neighborhood of every 8 pixels along X and Y coordinates of aim points, are used to smooth the aim point direction, and the mask size for smoothing point directions is 5×5. Thus, the smoothing area is more reasonable, the smoothing degree is stronger, and the result is better.

2.2 Symmetrical Axis of the Core Estimation

In addition, the orientation θ_c of the symmetrical axis of the region near the core is one of the main features of modal area. It can be used for fingerprint classification and taken into account for fingerprint matching in which the rotation angle is adjusted between the input image and the template image. Therefore, it is an important issue to compute the symmetrical axis of the region near the core reliably. Some literatures only discuss the case that the core point is in the upper section of the image, so it is variational under rotation. In this paper, the idea of V. S. Srinivasan et al.[20] is adopted and improved to compute the more accurate consecutive orientation rather than the coarser disperse orientation. The key idea is that the orientation θ_c is the statistical dominant direction among a coarsely chosen disperse orientations in the region of core point.

Based on the methods presented above, figure 1 shows some examples of pseudoridges extraction for typical fingerprints, and figure 2 shows some examples of symmetrical axis of the core estimation. In figure 1, the different color lines denote the results gained by different tracing ways (clockwise tracing or anti-clockwise tracing). And in figure 2, the black-white line denotes the correct orientation θ_c of the symmetrical axis of the region near the core, and the green line denotes the orientation which is determined by other factors (for example, the second maximum of the percentage for every template).

3 Fingerprint Classification Scheme

The Detection of singular points which is used in fingerprint classification has been reported in literature [21]. In this section, a refined fingerprint classification scheme is introduce as follows:

For the purpose of classification, the utilization of a global geometric feature to describe uniquely the general shape of fingerprint ridges within a particular class has been proposed. That each fingerprint class possesses a distinct geometric feature which is descriptive of the class' global ridge shape has been discovered. However, it is also observed that this global geometric feature of a particular class might also exist in another class. Nevertheless, it has been made a conjecture that this is due to a progression of the class ridge patterns from the simple to the complicated in which a distinct feature of a complicated or high class would not be found in a simple or lower class. Therefore，based on this understanding, in order to resolve the above feature ambiguity problem, operation in a top-down manner must be adopted [8], namely, the complicated or high class is determined in advance, secondly, the simple or lower class is determined. So, the order for fingerprint classification proposed in this paper is that the global geometric feature of the twin class is extracted firstly, and secondly the whorl class' is detected, then the other classes. The specific steps are as follows:

Firstly, the potential turns which are near the core points are found out from a set of the traced pseudoridges points above. The method is that the region, around which the distances between every vertices gained by tracing pseudoridges are very small and the number of vertices is considerable, is found out and the center point of this region is treated as the turn point.

Secondly, the global geometric framework (pseudoridge) traced is smoothed. The control vertices between which the distance is less than a threshold value are incorporated. Further, the above processed control vertices which are in a line on the whole and also in a predefined range are incorporated. After smoothing processing, the fake turning points would be reduced enormously and the subsequent computation is also cut down.

Finally, a set of vertices vectors which forms the turned ridges is found out from the set of vertices vectors in which the pseudoridge' corresponding control vertices are re-employed as approximating tangent vectors with each pair of vertices representing a vector in the direction of increasing indices. Assuming V to be the set of vertices vectors, a turn is made if

$$CosValue \quad < -0.908 \tag{1}$$

where,

$$CosValue = v_i \cdot v_j / |v_i| \cdot |v_j| \qquad v_i, v_j \in V \qquad and \qquad i \neq j \tag{2}$$

Then, the vertices vectors which make up the turns should be signed and used for the following fingerprint classification.

(1). Classification of the twin type

The twin type should be determined by the unique feature that the twin type takes the shape of two non-monotonic turns or turns with opposite signs. See Fig. 3, if the number of the detected turns is two or more than two and there is a set of sequence

vertices vectors which make up the turns: Va, Vb and Vc, where, Vc is the joint vector of two turns, and where the angle between the line from the center point of Vc to the center point of Va and the line from the central point of Vc to the central point of Vb is an obtuse angle, then the fingerprint is classified as a twin type. Otherwise, the next step should be carried through.

(2). Classification of the whorl type

In contrast with the twin type, a whorl ridge exhibits a spiral-like shape and takes the form of at least two monotonic turns or turns with similar signs. See figure 4, similarly, if the number of the detected turns is also two or more than two and there is a set of sequence vertices vectors which make up the turns: Va, Vb and Vc, where, Vc is the joint vector of two turns, and where the angle between the line from the central point of Vc to the central point of Va and the line from the central point of Vc to the central point of Vb is an acute angle, then the fingerprint is classified as a whorl type. Otherwise, the next step should be carried through.

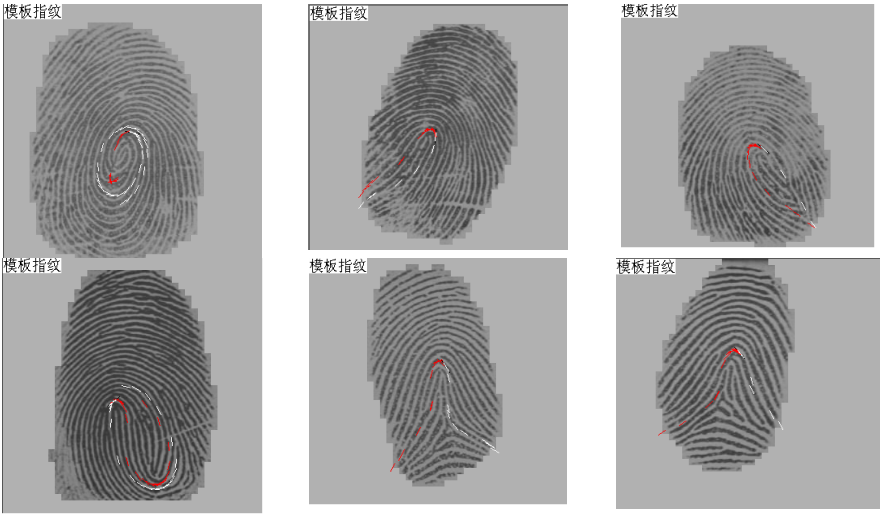


Fig. 1. Examples of pseudoridges extraction for typical fingerprints

(3). Classification of the loop type

If the number of turn is only one, then the core point should be checked by the turn point computed above. Namely, if the core point is near the turn point, then the core point is true, otherwise, the turn point replaces the core point. Subsequently, using the method presented above, the symmetrical axis of the region near the core is estimated. The position interrelation between the traced pseudoridge and the symmetrical axis of the core is determined. If the start point and the end point of pseudoridge are all on the left of the symmetrical axis, then the fingerprint is classified as a left loop type, if the start point and the end point of pseudoridge are all on the right of the symmetrical axis, then the fingerprint is classified as a right loop type.

In order to examine the case in which the pseudoridge's end point terminates on a side of the symmetrical axis (left or right) and forms another side's (right loop or left

loop) trend, if the two end points are on each side of the symmetrical axis respectively, then the direction angle between the vector with the start point as well as its neighborhood point of pseudoridge and the symmetrical axis, and the direction angle between the vector with the end point as well as its neighborhood point of pseudoridge and the symmetrical axis, are both computed. If both position relations constituted by this two direction angles form the left loop trend and the direction angles are larger than a threshold value, then the fingerprint is classified as a left loop type. Or if both position relations constituted by this two direction angles form the right loop trend and the direction angles are larger than a threshold value, then the fingerprint is classified as a right loop type. Otherwise, the fingerprint is classified as a tented arch type.

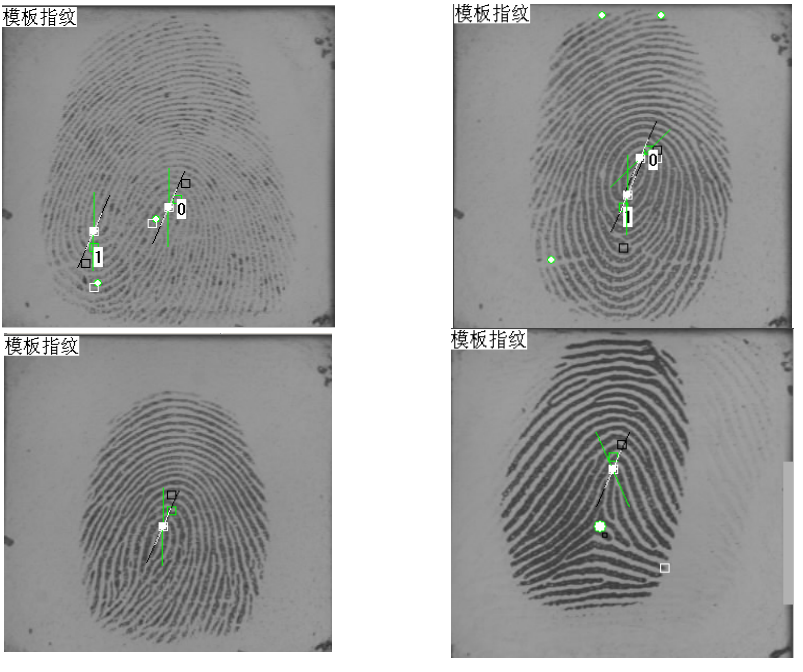


Fig. 2. Examples of the symmetrical axis of core

(4). Classification for the other cases

In other situations, the following cases should be considered:

A. If there are two core points or more than two core points, then a whorl type is assigned;

B. If there are a core point and a delta point, then judge the following conditions.

(ⅰ) If the angle, α, between the line segment from the core to the delta and the symmetric axis is less than a predefined threshold value, Θthreshold, then a tented arch type is identified;

(ⅱ) If α is more than Θthreshold and the delta point is on the left of the core's symmetric axis, then the fingerprint is classified as a right loop type;

(iii) If α is more than Θthreshold and the delta point is on the right of the core's symmetric axis, then the fingerprint is classified as a left loop type

C. If there are no core points and no delta points, or no turns in the traced pseudoridge, then a arch type is identified.

If the conclusion in the step (3) and the conclusion in the step B of (4) are same, then this conclusion is reserved, otherwise, the following steps should be carried through.

If the traced pseudoridge is mostly on the right of the line segment from the core to the delta, then the fingerprint is classified as a right loop type;

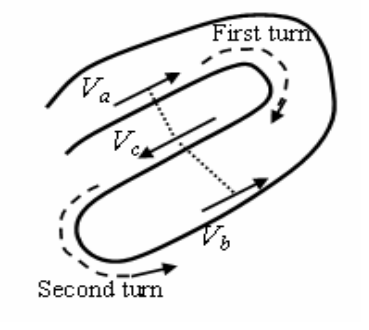


Fig. 3. Global geometric shape feature of the twin type

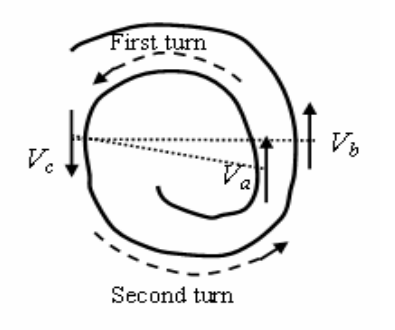


Fig. 4. Global geometric shape feature of the whorl type

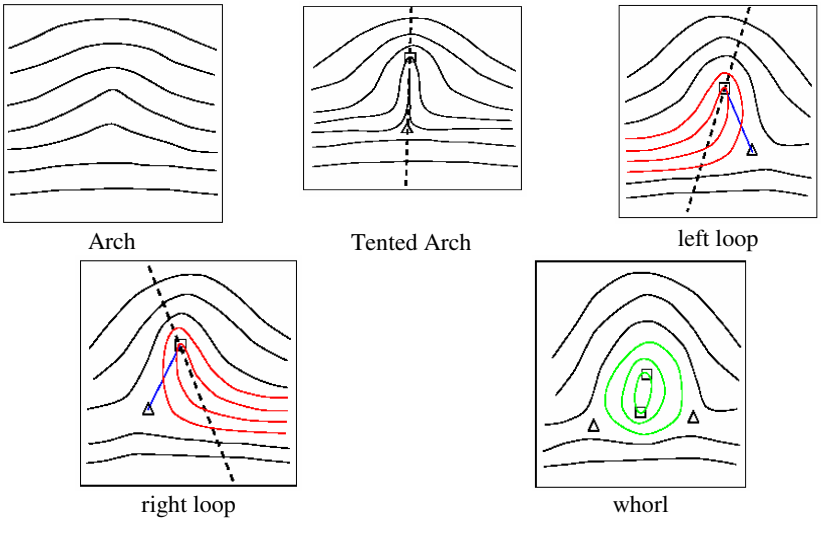


Fig. 5. Sketch maps of some fingerprint classes

If the traced pseudoridge is mostly on the left of the line segment from the core to the delta, then the fingerprint is classified as a left loop type;

Otherwise, the fingerprint is classified as a tented arch type.

If the conclusion in the step (2) and the conclusion in the step A of (4) are same, then this conclusion is reserved, otherwise, the conclusion in the step A of (4) is preferential.

Lastly, if none of the above conditions is satisfied, then the fingerprint is rejected.

4 Experimental Results and Conclusions

The classification algorithm described above has been tested on the 1000 typical fingerprints of NJU fingerprint database which contains 2500 images taken from 250 different fingers, 10 images per finger, and these fingerprint images are of varying quality in order to provide a realistic situation. The results are shown in table 1. The first column shows the class index, and the first row gives the assigned class using the current approach. The class index of one fingerprint in the database does not necessarily belong to only one class. According to the experimental results, the accuracy is 87% without rejection, if the tented arch and the arch are combined into one class, the accuracy rises to 92.2%. And a lower error rate can be achieved by adding the reject option based on the quality of the images.

Table 1. Experiment results tested on NJU fingerprint database

True Class	**Assigned Class**					
	Left loop	Right loop	Whorl	Tented Arch	Arch	Twin loop
Left loop	244	2	9	3	18	1
Right loop	3	259	7	1	11	2
Whorl	3	2	180	0	1	5
Tented Arch	4	5	1	96	45	0
Arch	0	2	1	2	69	0
Twin loop	2	0	8	0	0	78

The causes of mistake classification are that some images are of low quality due to noises, or that fingerprint images collection is not in its integrity, or that traced pseudoridges is wrong due to the geometric framework being around the central area where the modal area varies complicatedly. Meanwhile, our fingerprint database is not a special fingerprint classification database, so, our method should be tested and improved on the normal database for fingerprint classification (NIST-4 database) and that of the latent fingerprints.

At first glance, the fingerprint classification problem appears to be rather simple. But because of large intraclass and small interclass variations in global pattern configuration and poor quality of input images, the issue of fingerprint classification is still a real challenge.

Since the framework makes use of a global shape feature, it is less susceptible to the image noise, and combines but not completely uses the local feature such as core and delta points, the classification algorithm proposed in this paper also works well when false singular points exist or true singularities are missing. Moreover, our algorithm operation is irrelative to a fixed state of orientation and position, so it is invariant under transition and rotation. In the future, the focus will be put on improving the algorithm and investigating a practical fingerprint classification scheme which is based on the pattern similarity.

References

1. Kawagoe, M., Tojo, A.: Fingerprint pattern classification. Pattern Recognition 17(3), 295–303 (1984)
2. Sherlock, B.G., Monro, D.M.: A Model for Interpreting Fingerprint Topology. Pattern Recognition 26(7), 1047–1055 (1993)
3. Karu, K., Jain, A.K.: Fingerprint Classification. Pattern Recognition 29(3), 389–404 (1996)
4. Hong, L., Jain, A.K.: Classification of Fingerprint Images. In: 11th Scandinavian Conference on Image Analysis, Kangerlussuaq, Greenland (June 7-11, 1999)
5. Wilson, C.L., Candela, G.T., Watson, C.I.: Neural Network Fingerprint Classification. J. Artificial Neural Networks 1(2), 203–228 (1993)
6. Candela, G.T., Grother, P.J., Watson, C.I., Wilkinson, R.A., Wilson, C.L.: PCASYS: A Pattern-Level Classification Automation System for Fingerprints. NIST Tech. Report NISTIR 5647 (August 1995)
7. Senior, A.: Hidden Markov Model Fingerprint Classifier. In: Proceedings of the 31st Asilomar conference on Signals, Systems and Computers, pp. 306–310 (1997)
8. Chong, M.M.S., Ngee, T.H., Jun, L., Gay, R.K.L.: Geometric framework for Fingerprint Classification. Pattern Recognition 30(9), 1475–1488 (1997)
9. Maio, D., Maltoni, D.: A Structural Approach to Fingerprint Classification. In: Proc. 13th ICPR, Vienna, VC, pp. 578–585 (1996)
10. Fitz, P., Green, R.J.: Fingerprint Classification Using Hexagonal Fast Fourier Transform. Pattern Recognition 29(10), 1587–1597 (1996)
11. Hankley, W.J., Tou, J.T.: Automatic Fingerprint Interpretation and Classification via Contextual Analysis and Topological Coding. In: Cheng, G.C., et al. (eds.) Pictorial Pattern Recognition, pp. 411–456. Thompson Book Co. (1968)
12. Moayer, B., Fu, K.S.: A Syntactic Approach to Fingerprint Pattern Recognition. Pattern Recognition 7, 1–23 (1975)
13. Kameshwar Rao, C.V., Black, K.: Type Classification of Fingerprints: A Syntactic Approach. IEEE Trans. Pattern Anal. and Machine Intell. 2(3), 223–231 (1980)
14. Bowen, J.D.: The Home Office Automatic Fingerprint Pattern Classification Project. In: Proc. IEE Coll. on neural network for image processing applications (1992)
15. Hughes, P.A., Green, A.D.P.: The use of Neural Network for Fingerprint Classification. In: Proc. 2nd Int. Conf. on Neural Network, pp. 79–81 (1991)
16. Kamijo, M.: Classifying Fingerprint Images using Neural Network: Deriving the Classification State. In: Proc. 3rd Int. Conf. on Neural Network, pp. 1932–1937 (1993)
17. Moscinska, K., Tyma, G.: Neural Network based Fingerprint Classification. In: Proc. 3rd Int. Conf. on Neural Network, pp. 229–232 (1993)
18. Tan, T., Yu, Y., Cui, F.: A Robust Pseudoridges Extraction Algorithm for Fingerprints. In: Li, S.Z., Lai, J.-H., Tan, T., Feng, G.-C., Wang, Y. (eds.) SINOBIOMETRICS 2004. LNCS, vol. 3338, pp. 532–538. Springer, Heidelberg (2004)

19. Wang, L., Dai, M.: Localization of singular points in fingerprint images based on the Gaussian-Hermite moments. Journal of Software 17(2), 242–249 (2006)
20. Srinivasan, V.S., Murthy, N.N.: Detection of Singular Points in FingerPrint Images. Pattern Recognition 25(2), 139–153 (1992)
21. Tan, T.Z., Ning, X.B., Yin, Y.L., et al.: A Improved New Method For Detection of the Fingerprint Singularities. In: 4th Chinese National Conference on Biometric Recognition, Beijing, China, pp. 184–187 (2003)
22. Cappelli, R., Maio, D., Maltoni, D., Wayman, J.L., Jain, A.K.: Performance Evaluation of Fingerprint Verification Systems. IEEE Transactions on PAMI 28(1), 3–18 (2006)

Study on Embedded Vehicle Dynamic Location Navigation Supported by Network and Route Availability Model

Zhang Dong[1,2], Qian Depei[1], Liu Ailong[2], Chen Tao[2], and Yang Xuewei[2]

[1] School of Electronics and Information Engineering, Xian JiaoTong University, Xian, P.R. China
Navgrid@163.com
[2] Xian Research Institute of Surveying and Mapping, Xian, P.R. China

Abstract. The Route Availability (RA) in vehicle navigation is a primary metric to measure the usability of the vehicle navigator. Few researches have been reported on usability-based route analysis. Unlike common independent navigator, this paper constructed a dynamic mechanism upon on network communication chain supporting real time vehicle navigation, which includes road traffic information collection, data updating, dynamic information broadcasting, availability route programming. Concretely, the concept of programming Route Group (RG) is first introduced. Then, the concept of route segment and the availability metric of each segment are defined. Upon these definitions, a route availability measurement model using probability method is presented. Based on this model, a comprehensive metric, Z, for analyzing and comparing route availability is proposed to address the dynamic change in the road network. A new route programming algorithm called Z-algorithm is designed. Comparison of the Z-algorithm and traditional route programming algorithms such as the Dijkstra algorithm shows that the route obtained from the Z-algorithm is more applicable and satisfies usability requirement of vehicle navigation.

1 Introduction

The main purpose of the navigator is to lead drivers to their destination by voice or graphic guidance according to the computed route. The route is the major factor throughout the whole flow. The main aspects of performance evaluation for vehicle navigator include: flexibility in looking up the destination, map display rate and visual effect, accuracy and coverage of road net, route availability, and real-time responsiveness of route programming (Dong, Z., 2005), where the route availability, difficult to evaluate, is the most important metric.

There are about 20 representative types of different route programming algorithms used currently (Pierre-Yves, G., 2003). For calculating the shortest path between two points, Floyd's algorithm is used commonly with a time complexity of $O(n^3)$. The double Sweep Algorithm can resolve the top-k routes between one node and others. The Bellman-Ford-Moore Algorithm, though with a better time complexity, is not

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 713–722, 2007.

good in practice (Retcher, G., 2006). A* algorithm is the most widely used heuristic algorithm. Its main characteristic is that it uses the known information combined with the distance from the current point to the end in calculating the next point. In the searching period, the most possible next point is selected firstly in calculation for reducing calculation times and improving efficiency. In navigation systems, the Dijkstra algorithm, brought forward in 1959, has been considered as one of the most efficient algorithms for finding the shortest path so far (Jagadeesh G. R, 2002). The shortest distance between a node and all of others in a graph can be calculated with a time complexity of $O(n^2)$, in which n is the number of nodes.

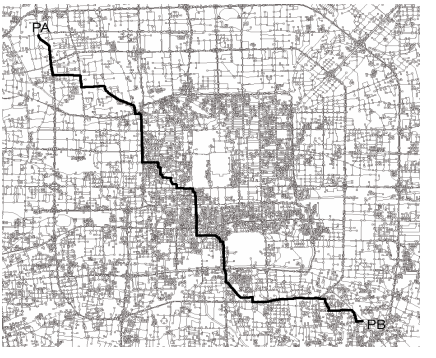


Fig. 1. Route by simple algorithms

In real navigation, the route just is a choice, and we can not evaluate whether it is good or not. Often, the computed route is not the best or indeed not applicable to guide the driver to the destination. Fig. 1 shows an experiment result came from Dijkstra algorithm, which is from the northwest corner to the southeast of the 3rd Ring Road in Beijing, China. Obviously the route is not applicable although the path computed is the shortest, because: (1) Cover the downtown area where the traffic is heavy to pass. (2) There are too many traffic restrictions and traffic lamps on the route. (3) A great number of segments of the route are lower classes with awful pass ability.

For numerous drivers, more applicable route is to go maybe with a longer distance. The experiment shows: (1) In navigation process, it is not enough only to give out a route to driver without availability. (2) We should study route availability concept, mathematical presentation, measurement model and application methods, etc. (3) It is necessary to obtain dynamic traffic conditions of a route segment at different time.

However, few systemic studies on these issues have been reported in the navigation field. In fact, navigation in unfamiliar environments is a common and demanding cognitive activity. Due to a series of errors, such as late lane changes, inappropriate traffic management, real time passing situation, etc, always driver can not get available route (May, A.J., 2003). Some navigation works with centre support was reported, but no dynamic information was used (Hunaiti Z., 2006). The study need for route availability include: availability definition, measurement model, effect

factors, modified algorithm, and corresponding availability experiments, etc. Therefore, we should analyze the route availability from applicable point of view.

2 Route Availability Model

2.1 Mathematical Representation of Route Availability

Definition 1. Programming Route Group (RG):

$$RG_k\ (PA, PB)=SET(RR_i) \qquad 1\le i\le k \tag{1}$$

Here, PA is the starting point, and PB is the ending point. Because of traffic restriction and distraction from the route, the route programming from PA to PB needs to be realized several times, presented as k times, and the actual route passed through by vehicle is a collection of k Real-time Routes (RR). RR_i denotes the i^{th} route, and k=1 denotes the vehicle following a single selected route from PA to PB. Regarding the rationality and availability of the i^{th} route, the analysis results show that some segments of the route are rational and practicable and the others are not, or difficult to pass through. So it is necessary to divide the route into segments as:

Definition 2. Segmentation of the route RR_i:

$$RR_i^g\ (PT, PB)=SET(RR_i^j) \qquad 1\le j\le g \tag{2}$$

Here, PT is the position where distracting from the selected route; RR_i^g indicates that RR_i is composed of g segments, RR_i^j denotes the j^{th} segment of RR_i, and g=1 denotes the route consisting of a single segment.

From the above definition, $RR_{k(i)}^{g(j)}$ (PA, PB) is used to denote that there are k programmed sub-routes from PA to PB in actual navigation, and the current i^{th} sub-route consists of g segments, and the current segment is the j^{th}.

Definition 3. Route Availability (RA):

$$RA\ (PA, PB) = D_{rn} / (D_{rn} + D_{rs}) \tag{3}$$

Based on the concepts of sub-route and segment, we can give out Route Availability presentation.

Here D_{rn} is the length for a vehicle to pass smoothly, D_{rs} is the length difficult to pass through because of lower class road, traffic jams, or temporary road conditions and traffic restriction, etc. Obviously, D_{rn} and D_{rs} are dynamically changing. For a segment, it is important to evaluate the pass smooth degree in a period of time, and the data mainly comes from experience data and traffic information broadcast system.

Accordingly, the availability of the sub-route RR_i can be defined as:

$$RA\ (RR_i) = D_{Irn} / (D_{Irn} + D_{Irs}) \tag{4}$$

Here D_{Irn} is the length of the route segment within the i^{th} sub-route on which the vehicle to pass normally, and D_{Irs} is the one that the vehicle passes abnormally. Route availability is related to user's demand, relevant algorithms, changing road net condition, traffic jam condition, other random events, etc.

2.2 Availability Measurement Based on Probability Model.

Definition 3 is only suitable for a statistical estimation of the route availability. In the real driving situation, availability measurement is much more complicated. The i^{th} sub-route will not be available if any RR_i^j of the segments can not be passed through. Here, RG (PA, PB)=$\{RR_1^{g(j)}, RR_2^{g(j)}, \ldots\ldots, RR_k^{g(j)}\}$, and $RR_i^{g(j)}=\{RR_i^1, RR_i^2, \ldots\ldots, RR_i^g\}$ are g segments of a sub-route RR_i.

The availability measurement is defined by the probability that the vehicle can pass a route normally, denoted approximately with the probability model. In order to perform the measurement, the user demand model and conjunction relation among segments need to be defined according to the previous availability model.

Definition 4: Conjunction of route segments:

$$C_i^j = C\ (RR_i^j, RR_i^{j+1}) \text{ or } C\ (RR_i^g, RR_{i+1}^1) \tag{5}$$

The conjunction relation among route segments, expressed as C, is actually represented by the connecting point between route segments, usually an intersection. Therefore, a route can be presented as a series of segments, separated by intersections. Let C_i^j be the conjunction between road segments j to j+1 in the sub-route i. When j=g, it denotes that the last segment of sub-route i is connected with the first segment of sub-route i+1.

This paper uses RBD model to analyze route RR availability as shown in Fig. 2, which is the most common sub-system partitioning method used in system availability and credibility analysis (Sathaye, A., 2000, Balkovich, E., 1987) .

$$C_1^1 - RR_1^1 - C_1^2 - RR_1^2 - - - - C_i^j - RR_i^j - C_i^{j+1}$$

Fig. 2. Virtual RBD model for real-time route partition

Here, route segment RR_i^j and connection relation C_i^j are independent. Suppose the abnormal-passing probability of route segment RR_i^j is $f_s\ (RR_i^j)$, and that of conjunction C_i^j is $f_r\ (C_i^j)$. Then, the route availability based on probability model can be presented as:

$$RA(PA, PB) = \prod_{i=1,j=1}^{k,g} (1- f_r (C_i^j)) \cdot \prod_{i=1,j=1}^{k,g} (1-f_s (RR_i^j)) \quad (6)$$

The acquiring mechanism of $f_r(C_i^j)$ and $f_s(RR_i^j)$ is complicated. Similar to D_{rn} and D_{rs} in Definition 3, the probability data comes from: (1) Integrated experience data of $f_r^{'}(C_i^j)$ and $f_s^{'}(RR_i^j)$ based on the historical statistical data in certain period of time. (2) Traffic condition information broadcasting systems. The navigator can receive real-time traffic jam information and adjust the weights of the road segments dynamically to ensure availability of the programmed route.

3 Dynamic Information Acquisition and Broadcast Based on Network

3.1 Dynamic Information Network Flow

Fig.3 shows dynamic route programming and navigation framework, unlike independent navigator without any real time information supporting. Here, there are three parts: Unit I, Unit II, and Unit III which communicate each other via the channel following GSM or CDMA protocol.

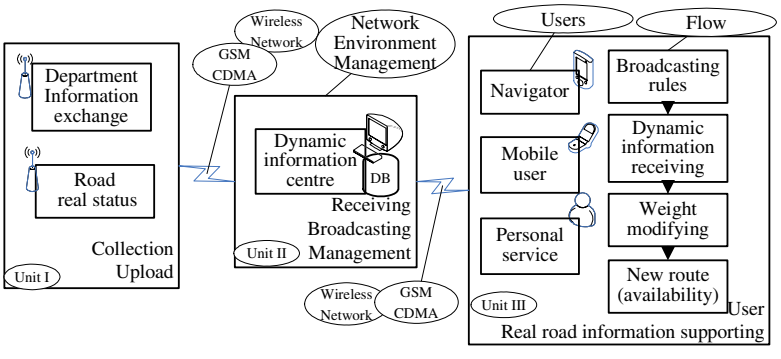


Fig. 3. Dynamic navigation mechanism based on network environment

Unit I achieves dynamic traffic collection and uploading. The information mainly comes from two ways. Regular information exchange among special departments is secondary, because the situations are governable and not frequent such as Road segment in Construction, compulsive traffic Forbidden Rules, road Close via Whether reasons or traffic Accident, respectively presented as { RiC,FR, CiW,CiA}, etc. The main information resources with obvious dynamic characteristics are collected through vehicle uploading. Primary comments are Current discrete Status of present road segment of vehicle defined as CS ranking as discrete three levels, fluent passing, weak-jam and strong-jam.

Unit II is dynamic information management centre for data receiving, data management, and broadcasting.

Unit III is user client to meet real vehicle navigation based on dynamic road traffic data. For any application, dynamic information receiving in term of broadcasting rules, weight modifying and route programming with availability form the integrate dynamic flow.

3.2 Dynamic Information Acquisition Chain

Dynamic information includes data set {RiC, FR, CiW, CiA} and CS. {RiC, FR, CiW, CiA} can be announced by special management departments such as offices of communications. The collection of CS is complicated correspondingly.

The variable CS is evaluated as 1, 2, or 3, which mean fluent passing, weak-jam, or strong-jam. In fact, only the situation of CS=3 possibly can be reported to the centre through a simple program button on the screen. The experience data combined with mobile map data to ensure availability computation.

Frame format，meeting GSM or CDMA communication protocol，is defined as:

$CS, FType, <DATA> * hh <CR> <LF>.

Here, " $CS" denotes frame head flag; " FType" denotes frame type of 5 bytes ASCII code; <DATA> is binary data segment composed with Struct CSpackage, every character is from 0x30 to 0x3F; "*" means interval symbol; "hh" is check sum; <CR><LF> present enter and new line.

The final experience data will be transformed to the abnormal-passing probability $f_s\ (RR_i^j)$ of route segment RR_i^j, and the $f_r\ (C_i^j)$ of conjunction C_i^j from {RiC, FR, CiW, CiA} and CS, in a certain period of time. The ways of farther step, transforming the $f_s\ (RR_i^j)$ and $f_r\ (C_i^j)$ to road weight and modified road length employed in route programming algorithm directly, will be shown in section 4.

3.3 Real-Time Information Broadcasting.

Traffic information broadcast system provides real time traffic jam information or other road conditions. The broadcasting package is similar to CSpackage.

The data broadcasting adopts similar communication protocol format with data uploading. Frame format for broadcasting is defined analogously as:

$ED, FType, <DATA> * hh <CR> <LF>.

Here, <DATA> is binary data segment composed with Struct EDpackage. The broadcasting rule is very important because of great amount of vehicles and complicated road situations. Experiment shows no system barrage arise when parallel 1000 vehicles is online synchronously.

4 Dynamic Route Programming Supported by Network and Availability Metric

Dynamic route programming mainly includes the programming algorithm supported by dynamic traffic information to satisfy availability demands.

4.1 Dynamic Route Programming Based on Availability

The route programming algorithm involves several elements, denoted as F(PA, PB, CA, RND, RCT, RCI, RTR) and $f_s(RR_i^j)$, $f_r(C_i^j)$. Here, CA, RND, RCT, RCI, RTR denote current point, road net data, road classification table, crossing information, route topological relationship, respectively, $f_s(RR_i^j)$ and $f_r(C_i^j)$ denote dynamic passing information of route segment RR_i^j and that of conjunction C_i^j. At the same time, real route programming can take several modes, I={I1, I2, I3, I4, I5}, corresponding to distance priority, normal pass priority, less toll priority, expressway priority, and time priority, respectively. One route programming can involve one mode or combination of several modes. Each involved mode has a corresponding weight, so the weight vector R={R1, R2, R3, R4, R5 } reflects the influence of each mode. The combination of modes, RA', is represented as:

$$RA' = I \cdot [R1, R2, R3, R4, R5]^T \tag{7}$$

The distance L is the major criterion when selecting the next node in route programming. In real computation, the L should be adjusted according to L-Scale which is determined by the route programming modes and the dynamic information of availability demands. Actually, L-Scale is a changing factor of distance L in real computation.

$$\text{L-Scale} = RA' \oplus \left(\prod_{i=1,j=1}^{k,g} (1 - f_r(C_i^j)) \cdot \prod_{i=1,j=1}^{k,g} (1 - f_s(RR_i^j)) \right) \tag{8}$$

Here, '$\oplus$' is an integrated computation operator, maybe include comparison, logical operation, plus, etc., and takes consideration of both factors: the routes programming mode RA' such as distance priority, time priority, etc., and the route segment availability metric RA (CA, PB). RA (CA, PB) is a special data set updated dynamically with the time and can be accessed through the link to the route segment ID mentioned above.

In actual computation, we replace the distance comparison factor L with integrated comparison metric Z, and Z is then represented as:

$$Z = L^* (\text{L-Scale}) \tag{9}$$

Here: when $f_s(RR_i^j) < 15\%$, $Z = L*100\%$; when $f_s(RR_i^j) > 85\%$, $Z = \infty$.

4.2 Integrated Z-Algorithm

An improved algorithm, Z-algorithm, based on the above dynamic mechanism and integrated metric Z is shown as:

[Z-algorithm]

```
Input: {PA,PB,CA,RND,RCT,RCI,RTR,f_s(RR_i^j),f_r(C_i^j)}
Output•Real-time programming Route: RR
Begin•{
    InputDestination(CString &PA,&PB);
    GetPosition(Cstring &CA); //Current position;
    RouteProgramming:
    ConstructNodeSet(Struct &S1,&S2); //Set up a
    temporary node set S1 and a permanent node set S2;
    S1=null; S2={all nodes};
    BroadcastJudgement(Bool &SegofRoad);//Segment holds
    relative dynamic information or not;
    ReceivefromCentre(Struct &EDpackage); //Dynamic
    information acquisition;
    L1:
      SearchNode(Cstring &P1 from S2);//A new extended
      node P1 from S2;
      ConstructMetric(long* Z); //Availability metric
      Z=L(P1)*(L-Scale(P1)) with received data;
      If(P1∈S1)and(Z<value of P1 in S1) then Replace
      P1; //Replace P1 if it has existed in S1 with
      smaller Z;
      If(P1∈S2) then Delete P1;//Delete P1 if it is
      inS2;
      If P2=null then RR=S1; //Get route RR;
      else  goto L1;
    …… }
End
```

5 Experiment and Analysis

The experiments are conducted by means of an embedded navigator Nav-1 which was developed with above algorithms by our lab. Nav-1 consists of a touch screen with a 320X240 resolution, audio output, 64MB SDRAM, and a 512MB CF storage. The CPU is a 206MHz Inter Strong ARM SA1110 processor, powered by 12V DC. It is 198mm in width, 122mm in height and 37.6mm in thickness, and weights 600g.

The Dijkstra algorithm is suitable for single programming mode, such as R1=1 or R5=1. The route RR1 (as shown in Fig.1) programmed goes along the diagonal direction with a shorter distance when R={1,0,0,0,0}. Z-algorithm can be used for mixed modes to meet user's multi-demands. The route RR' (as shown in Fig.4) under R={0,1,0,0,1} goes along the circle road with the higher class, but the distance is larger than RR1. In the progress, the vehicle received the dynamic, that the road of ID 34512 was in strong traffic jam, and another new route RR2 was programmed real time which is more available.

The first navigator using the Dijkstra algorithm generated the route RR1 and the Z-algorithm generated RR'-RR2. RR2 is the late route. Obviously, RR1 and RR'<RR2,

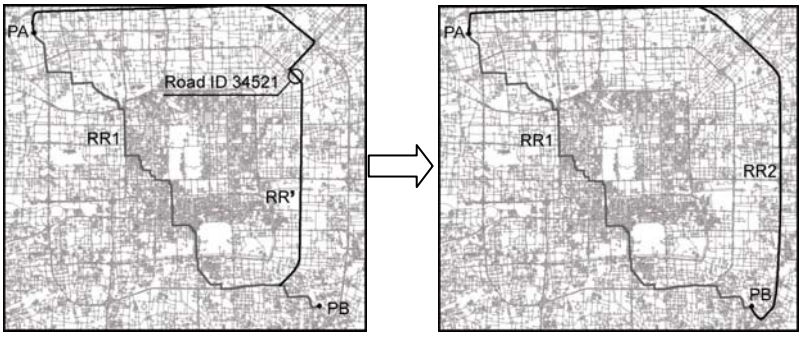


Fig. 4. Route produced by Z-algorithm

but we can find that RR1 and RR' are not available and RR2 is suitable to pass smoothly for numerous user.

In the experiment, the route was re-programmed ten times when driving along RR1. The second car using our Z-algorithm was able to reach the destination along RR2 with one route re-programming and 45 minutes ahead of the arrival of first car.

6 Conclusion

This paper has researched a new dynamic navigation mechanism, including dynamic route programming satisfying availability demands supported by dynamic real time traffic information. The acquisition information, came from two different ways, can be effectively transferred to centre, and transformed to statistical experience data stored in updating DB. According to special rules and protocol, the dynamic passing information can be broadcasted to user client and dynamic navigation is realized. From our experiments, we have learned that the route availability model proposed in this paper is rational and applicable. The method presented in this paper for measuring road availability is practical. Our modified Z-algorithm, based on the road availability and real time navigation, achieves a better navigation performance for smooth passing in real situation and shows better computation efficiency. But we have to know that road availability is only one metric of rationality. The rationality issue is more complicated and involves more factors which have to be taken into consideration. Our future research will focus on models and algorithms with more factors in real route navigation, such as segments availability data collection and management.

References

1. May, A.J., Ross, T., Bayer, S.H.: Drivers' Information Requirements when Navigation in an Urban Environment. The Journal of Navigation 56, 89–100 (2003)
2. Sathaye, A., Ramanni, S., Trivedi, K.S.: Availability Models in Practice. In: FTCC-1. Proceedings of the Int. Workshop on Fault-Tolerant Control and Computing, Shoel, Korea, pp. 823–829 (2000)

3. Balkovich, E., Bhabhalia, P., Dunnington, W., Wyant, T.: Vaxcluster Availability Modeling. Digital Technical Journal 5, 69–79 (1987)
4. Gilliéron, P.-Y., Konnen, J.: Enhanced Navigation System for Road Thematics. In: Proceedings of the 3rd Swiss Transport Conference, Switzerland (2003)
5. Retcher, G., Kealy, A.: Ubiquitous Positioning Technologies for Modern Intelligent Navigation Systems. The Journal of Navigation 59, 91–103 (2006)
6. Hunaiti, Z., Garaj, V., Balachandran, W.: A Remote Vision Guidance System for Visually Impaired Pedestrians. The Journal of Navigation 59, 497–504 (2006)
7. Jagadeesh, G.R., Heuristic, S.T.: Techniques for Accelerating Hierarchical Routing on Road Networks. IEEE Transactions on intelligent Transportation Systems 3(4), 301–309 (2002)
8. Dong, Z., Ailong, L., De, Z.: The Present Situation of Vehicle Navigation Techniques in China and the Multi-Topological Formation and Route Computing of National Navigation Map. In: Proceedings of the XXII International Cartographic Conference, La Cruna, Spain, pp. 1733–1738 (2005)

Convolution Filter Based Pencil Drawing and Its Implementation on GPU

Dang-en Xie[1], Yang Zhao[2], Dan Xu[1,*], and Xiaochuan Yang[3]

[1] School of Information Science & Engineering, Yunnan University 650091, China
[2] School of Information Science, Yunnan Normal University 650092, China
[3] South China University of Technology 510641, China
Tel.: +86-871-5737873 Fax: +86-871-5737873
danxu@vip.sina.com, xde820@gmail.com, xcy1198@163.com

Abstract. Traditional pencil drawing methods have their own drawbacks, such as modeling complexity and higher time-consuming. Thus, they are difficult to be suitable for the real-time applications. In the paper, we present a new pencil texture generating method based on the pencil filter. The method can conveniently generate the pencil drawing effect by convoluting the input image with the pencil filter. Moreover, the method is implemented on GPU, and then satisfies the requirement of real-time synthesis. Optical flow technique is used to guarantee the interframe coherence in video stylization.

Keywords: pencil filter, pencil drawings, Graphics Processing Unit (GPU), optical flow, non-photorealistic rendering.

1 Introduction

In the past decade, researchers in computer graphics community began to simulate traditional artistic media and styles, such as paintings [1], watercolor [2, 4], charcoal rendering [3, 5]. This is a new technique called Non-photorealistic rendering (NPR). Its purpose is not to aspire to the photorealism but to simulate the artist's work and represent the artistry, even the drawbacks of the artwork. To some extent, NPR is the complementarity of the photorealistic rendering.

Pencil drawing rendering is an important branch of NPR, which is firstly presented in the 90s last century [7, 8, 9, 10]. A key step of pencil drawing rendering is how to simulate the pencil texture. Sousa [8] attempted to model the physical behavior of pencil, paper and eraser. Their approach attained the pencil texture vividly. Later, Mao [6] simulates pencil texture using the line integral convolution (LIC) method. Also, they gain satisfied effect. LIC is a texture based vector field visualization technique which was first presented by Cabral and Leedom in 1993[10]. Given a 2D vector field represented as a regular Cartesian grid, the LIC algorithm takes as input a white noise image of the same size as the vector field and generates an output image wherein the texture has been locally blurred in the direction of the vector field.

* Corresponding author.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 723–732, 2007.

Although the traditional pencil texture generating methods obtain good effect, they all have disadvantages of deficiency and time-consuming. Physical modeling is very complexity. LIC method needs to calculate the visualization vector field of the input image, and then convolute the pixel one by one, so it also costs much time. Generally, using the LIC method to generate a pencil drawing needs about 20 minutes (here the image size is 1024*768) [10]. Both of the methods are not suitable to the real-time applications.

In this paper, we present a new method for simulating pencil texture by pre-calculating a special convolution filter, named pencil filter. It may have different appearances according to different stroke orientations and different stroke sizes. Once the pencil filters are made ready, we convolute the pixel of the black noise image with each corresponding filter, and then the pencil drawing image is accomplished. The proposed approach saves lots of time because pencil filters are generated in advance and the convolution operation is more efficient than traditional methods.

Obviously, the method can be extended to process video. To preserve interframe coherence of a video segment, the optical flow technique is adopted, which will be described in detail in Section 4. Additional, for real-time stylization applications, the paper performs an achievement of the method on GPU (Graphics Processing Unit).

2 Image Based Pencil Drawing

All existing pencil drawing techniques can be classified primarily into two kinds: geometry-based and image based. Geometry-based techniques take 3D scene descriptions as their input. Image based techniques directly process 2D images to get pencil drawing expressions. Our method belongs to the latter. Figure 1 shows the framework of our pencil drawing algorithm. Each processing box corresponds to a step of the algorithm:

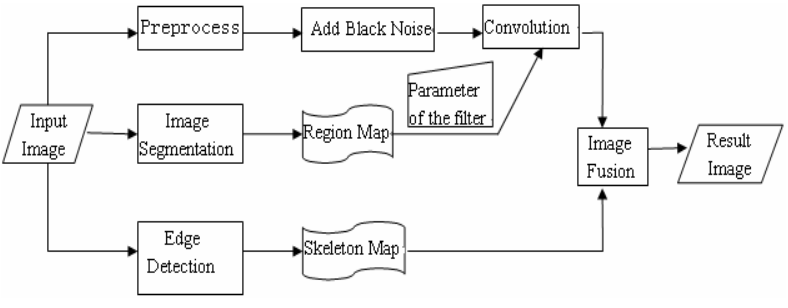
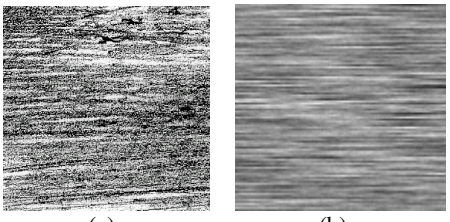


Fig. 1. The frame map of the algorithm

2.1 Generating the Pencil Filter

By observing and analyzing the real pencil texture (see Figure 2(a)), we simply suppose that: 1) graphite marks present stochastic distribution according to the coarseness of papers; 2) graphite marks stretch along the stroke tracks; 3) graphite marks on perpendicular direction of a stroke present obviously black-white staggered distribution.



(a) (b)

Fig. 2. Comparison of the real pencil texture (a) with the pencil filter generated texture (b)

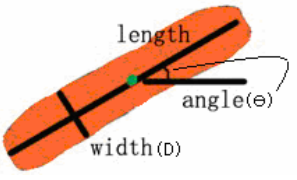


Fig. 3. Properties of the pencil stroke model

Considering the above supposes, we create a mathematic model for pencil filter. Assume that the stroke length is *len*, the stroke direction isθ and the stroke width is *2D*. As shown in Figure 3, if we know the stroke length and the stroke orientation, we can easily calculate the template size by $(\lceil len * \sin\theta \rceil \times \lceil len * \cos\theta \rceil)$.

The next problem is how to decide the value of each element in the pencil filter. As shown in Figure 4, firstly, calculate the distance *d* from each point *P* to the central axis *l* of a stroke. Then calculate the distance *r* from the point *P* to the center *O* of the stroke. The value of each element in the pencil filter lies on the relation between *d* and *D*, and also the relation between *r* and *len/2*.

Here, we take the upper right quarter of the template as an example (Figure 4(a)). Obviously, only three kinds of points are presented in the template. Points in the green area (e.g. P in Figure 4(b)) satisfy the conditions that *r* is less than *len/2* and *d* is

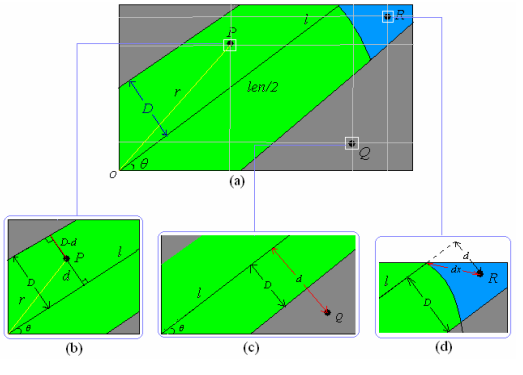


Fig. 4. Define a pencil filter

less than D, so we choose D-d as their values. Points in the gray area (e.g. Q in Figure 4(c)) satisfy the condition that d is large than D, which means the stroke can't covered this area, so their values are set to be zero. Points in the blue area (e.g. R in Figure 4(d)), satisfy the conditions that r is large than $len/2$ and d is less than D. Blue area is near to the stroke end, and the graphite marks is thin there, so the value in the template is less than D-d and none zero. We calculate the distance dx from point R to the end of line l, and then we choose D-dx as the value of this area. In this way, it decreases the value of the stroke end effectively, and the decrement is in proportion to the distance r. When D-dx is less than zero, the value should be set to zero.

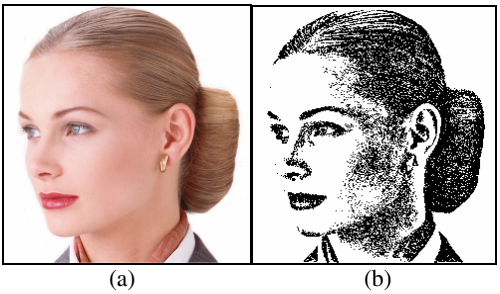
Viewing the pencil filter generating procedure, it properly simulates the real pencil texture:

1. The points near the stroke central line l have greater values. The larger the distance, the smaller the value. That is in accord with the real pencil texture property.
2. When the point beyond the stroke width, the value is set to zero. It insures the pencil stroke width;
3. The filter kindly simulates the stroke ends' physical property—graphite marks tapered with the disappear of the pressure on papers;
4. The points which have the equal distance to the central axis line have the equal value in the template. It insures the strength direction along the stroke.

2.2 Generating the Black Noise Image

To make sure the pencil texture has stochastic distribution, we generate the black noise image from the reference image. Our method for generating the black noise image is similar with the method for adding white noise in Mao [10]. We use the tone of the input image to guide the distribution of the black noise. Let I_{input} be the intensity of a pixel in the input image, P is a floating-point number generated with a pseudo-random function, and then the intensity I_{input} of the corresponding pixel in the noise image is decided in the following way:

$$I_{noise} = \begin{cases} 255, & if \quad P \le T \qquad P \in [0.0, 1.0] \\ 0, & otherwise \end{cases}, \quad T = k \cdot \left(\frac{I_{input}}{255} \right), \quad k \in (0.0, 1.0], \tag{1}$$



(a) (b)

Fig. 5. The black noise image ((a) is the original image; (b) is the corresponding black noise image)

in which k is a coefficient for controlling the density of the black noise. In this way we can promise the pencil drawings have the stochastic distribution character, and also promise the density of the black noise correspond to the intensity of the input image. If the intensity of the current pixel is lower, the value of T is smaller, and the probability of P large than T is larger, so the value of I_{noise} has more probability to be 0. On the contrary, the output value I_{noise} is 255. Figure 5(b) is the black noise image generated from the input image shown in Figure 5(a).

2.3 Extract the Contour Lines

A simplest artistic expression is extruding the outlines of the artwork. It is also an important step for generating the pencil drawing. In computer, we need to extract the contour lines for simulating the action of extruding outlines.

Gradient operators, such as Sobel, Robert, Prewitt and Kirsch operators are commonly used in digital image processing to extract edges of an image. Considering Kirsch operator [11] has the bigger weighted factors, we prefer to choose the Kirsch gradient operator to extract the contour lines in this paper, so that we can obtain the contour lines clearly.

The Kirsch operator has 8 filters. Formally, the Kirsch operator is defined by:

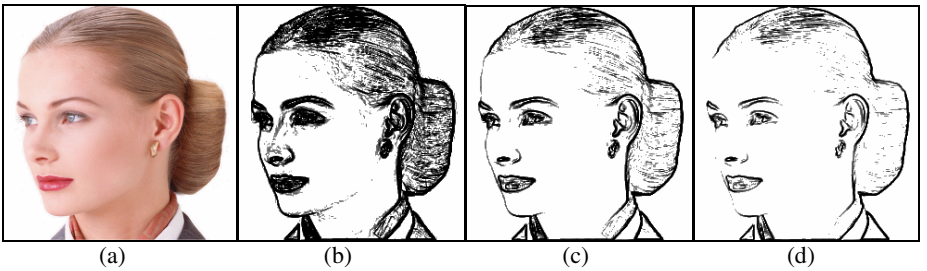
$$K(x, y) = \max\{1, \max\{5S_i - 3T_i\}\}, i = 0 \cdots 7 \tag{2}$$

in which,

$$S_i = f(A_i) + f(A_{i+1}) + f(A_{i+2}), \tag{3}$$

$$f(A_i) \text{ stands for the pixel value on position } A_i$$

$$T_i = f(A_{i+3}) + f(A_{i+4}) + f(A_{i+5}) + f(A_{i+6}) + f(A_{i+7}) \tag{4}$$



(a) (b) (c) (d)

Fig. 6. The contour lines maps with different value of μ

In practice, we are used to change the values of the Kirsch operator according to different images. Let K_i be the filter of the Kirsch operator, then we have:

$$K_i = \mu * K_i, i = 0 \cdots 7, \mu \in (0,1] \tag{5}$$

Here, μ is a coefficient for controlling the weight value in the filter. One can adjust the value of μ interactively, but we suggest that the value of μ should between 0

and 1. Figure 6 shows the different contour line maps with the different value of μ. Figure 6(b), (c) and (d) are show the results that μ is 1, 0.5 and 0.3, respectively. Generally, if an image has more details, the value of μ should be smaller. A smaller μ can prevent the contour line conglutination. On the contrary, if an image has little details, a larger μ should be set in order to make sure the consistency of the contour lines.

3 Implementation on GPU

In this paper, we transplant the pencil filter based stylization algorithm onto GPU. With the GPU's powerful parallel processing ability, we have achieved the real-time video synthesis for pencil drawing style. In this section, we will describe the GPU implementation of our algorithm in detail.

3.1 Convolution on GPU

It is difficult to generate pencil filter directly on GPU because the instructions and the registers are limited in GPU. Fortunately, we can generate the weight values of pencil filter in advance (like what shown in Figure 7). This idea makes a way to using our method on GPU. Firstly we load the weight values of pencil filter into GPU, which is generated in CPU in advance, and then convolution is executed for each pixel.

0	0.1509	0.3767
0.1509	0.6431	0.1509
0.3767	0.1509	0

0	0.0000	0.3293
0.3293	0.6827	0.3293
0.3293	0.0000	0

0	0	0	0.1001	0.0608
0	0	0.1038	0.3543	0.1001
0	0.1038	0.3543	0.1038	0
0.1001	0.3543	0.1038	0	0
0.0608	0.1001	0	0	0

Fig. 7. pencil filters using our method generated with the template parameters (left: *D*=1, *len*=3, θ=45°; middle: *D*=1, *len*=3, θ=30°; right: *D*=1, *len*=5, θ=45°)

Many effective means can achieve the convolution on GPU. Generally, we can store the weight values of the convolution template as a constant or uniforms type, and then execute the convolution operation. This method seems simple and feasible, but the fact is that if we solidify these constants into GPU's shader program, we can not easily expand the program function later on. It's not suitable to our method because we need to use the weight values to simulate the different property of the pencil stroke, such as stroke length, stoke width and stroke orientation. To solve the problem, we load the template data as a texture image, and call the image as Filter Texture. When a video fragment is synthesized in real-time, the corresponding filter texture will be chosen according to the user's need, and then convolution is executed. The convolution process can be expressed by:

$$filtered\ color = \frac{\sum k(s-s_0,t-t_0)tex(s,t)}{\sum k(s-s_0,t-t_0)} \tag{6}$$

where the current pixel position is (s0, t0), tex is a function for searching the corresponding filter texture. To save the memory, we store the total value of the template on the alpha channel.

3.2 Generating the Black Noise Image on GPU

At present, there's no function supplied for generating the floating-point pseudo-random number in GPU. Therefore, we generate a noise texture image in advance; the pixel value of the noise image is making up of floating-point random number. GPU can thereby get the floating-point random number by sampling the noise texture image. Then, we can generate the black noise image on GPU according to the approach described in section 2.2.

4 Stylization for Video Segment

The method described in Section 2, also can be used for rendering video segments. First, capture each frame from the input video. Then, use the same way to process each frame like what is used to deal with a single picture. Finally, rebuild the video with the processed frames. This procedure usually brings a negative effect to the result video because it cannot preserve the interframe coherence.

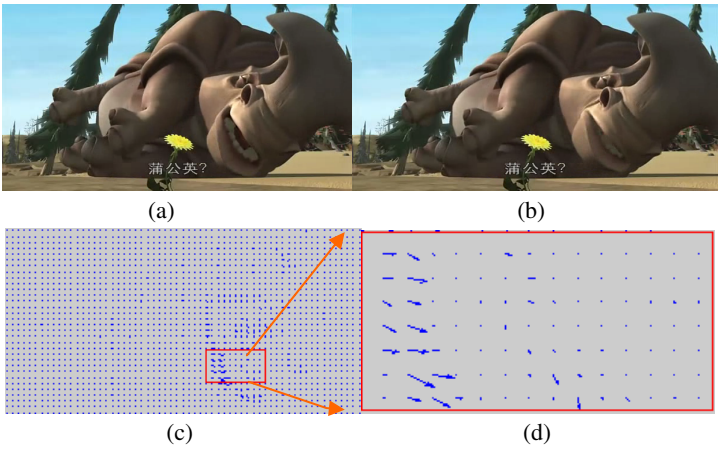


Fig. 8. Calculate the optical flow field((a) (b) are two adjacent frames. (c) is the optical flow vector field.(d) is the enlarged image of the red region of (c).)

To solve the problem, we use Horn's method [12] to estimate the optical flow field of each two adjacent frames. Figure 9 shows the optical flow vector field. As synthesis a frame, we compare the magnitude of the current pixel with an appointed threshold. If the former is small, then we believe this pixel is almost still. So we need only to copy the previous frame's corresponding pixel to the current pixel. Otherwise, we recalculate the value of the current pixel following with the method described in Section 2. Generally, the scene's change is very small in a pair of adjacent frames. So the

magnitudes of optical flow vectors are often to be zero or very small on most pixel positions, especially in the background. In this way, we can basically keep the interframe coherence of the video.

5 Experiment Result

A pencil drawing generating system on Windows environment has been built with the Matlab tool. Basically when the input image is specified, the system can generate the pencil drawing picture automatically. Users are allowed to specify some parameters interactively. These parameters control the stroke orientation, the stroke length, the density of the black points and the coefficients of the Kirsch operator. Figure 9 are some results generated by our method for static pictures. Figure 10 shows some video frames of real-time rendering results on GPU. In existing system, we do not allow the users to change the parameters during process a video stream. Instead, we set the default values for parameters in advance. Thus, the result quality is not good as static pictures because the default values are not usually suitable for all the frames.

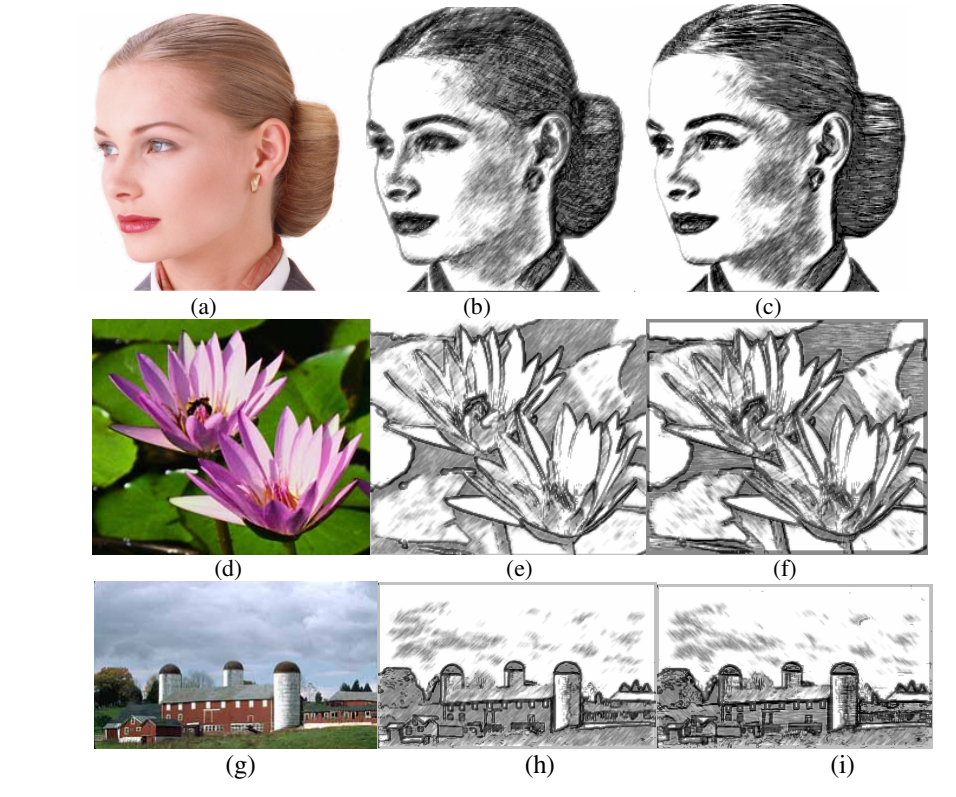


Fig. 9. Experiment results on CPU(figure(a),(d),(g) are original images; figure (b),(e),(h)are the pencil drawing images with single stroke orientation 45^0;figure (c),(f),(i) are the pencil drawing images with different stroke orientations.

Compared with the LIC method, our method saves lots of time. The primary reason is our method needn't to calculate the visualization vector field of the input image, and also needn't to do the hundreds of iterations. Our method only needs to do the convolution once for each pixel, and the kernel operator elements (Figure 7) usually have many zeros.
The system is developed using Matlab7.04. All experiments run on a 1.73GHz Pentium PC with 512M RAM. Cg(C for graphics) language is used for GPU programming.

Fig. 10. Real-time rendering result on GPU for some frames of the movie *Ice Age*

6 Conclusion

This paper presents a simple and efficient method for simulating pencil texture. Using the method, we accomplished an image based pencil drawing algorithm. Also, the method is successfully implemented on GPU to support real-time video stylization. The proposed method has the following advantages: (1) Efficient. The time cost for rendering a 1024*768 image on Matlab is 43.26 seconds; it is far less than LIC method, which needs about 20 minutes. (2) Convenient. Only single convolution is needed for pencil drawing image synthesis. (3) Bring a new way for real-time synthesis. It brings a significant reference for the other computer hardware, such as FPGA.

Acknowledgement

This work is supported by NSFC (No. 60663010) and NSF (No. 2006F0017M) of Yunnan province. All images are downloaded from the Internet. The video fragments are captured from the movie of "Ice Age".

References

1. Hertzmann, A.: Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In: SIGGRAPH 1998 conference proceedings, pp. 453–460 (1998)
2. Laerhoven, T.V., Reeth, F.V.: Real-time simulation of watery paint. Computer Animation and Virtual Worlds 16, 3–4, 429–439 (2005)

3. Lee, H., Kwon, S., Lee, S.: Real-Time Pencil Rendering. In: Proc. of the 4th Intl' Symposium on Non-Photorealistic Animation and Rendering, pp. 37–45 (2006)
4. Luft, T., Deussen, O.: Interactive watercolor animations. In: Proc. Pacific Graphics 2005, pp. 7–9 (2005)
5. Majumder, A., Gopi, M.: Hardware accelerated real time charcoal rendering. In: Proc. NPAR 2002, pp. 59–66 (2002)
6. Mao, X., Nagasaka, Y., Imamiya, A.: Automatic Generation of Pencil Drawing from 2D Images Using Line Integral Convolution. In: Proceedings of the Senventh International Conference on Computer Aided Design and Computer Graphics CAD/GRAPHICS 2001, pp. 240–248 (2001)
7. Takagi, S., Fujishiro, I., Nakajima, M.: Volumetric modeling of colored pencil drawing. In: Pacific Graphics 1999 conference proceedings, pp. 250–258 (1999)
8. Sousa, M.C., Buchanan, J.W.: Observational Model of Blenders and Erasers in Computer-Generated Pencil Rendering. In: Graphics Interface 1999 conference proceedings, pp. 157–166 (1999)
9. Sousa, M.C., Buchanan, J.W.: Computer-Generated Graphite Pencil Rendering of 3D Polygonal Models. In: EUROGRAPHICS 1999 conference proceedings, pp. 195–207 (1999)
10. Cabral, B., Leedom, C.: Imaging Vector Field Using Line Integral Convolution. In: SIGGRAPH 1993 conference Proceeding, pp. 263–270 (1993)
11. Castleman, K.R.: Digital Image Processing, pp. 390–391. Publishing House of Electronics Industry, Beijing (2002)
12. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artificial Intelligence 17, 185–203 (1981)

Improved LLE Algorithm for Motion Analysis*

Honggui Li[1] and Xingguo Li[2]

[1] Electronic Department, Physics College, Yangzhou University, Yangzhou 225002, China
hgli@yzu.edu.cn
[2] Department of Electronic Engineering, Nanjing University of Science & Technology, Nanjing 210094, China
xgli@njust.edu.cn

Abstract. Improved LLE algorithm is proposed. Usually there is translation in practical data. The form of translation relies on the type of data. Data alignment is a very important step for linear or nonlinear dimensionality reduction methods. Original LLE algorithm uses Euclidean distance to find neighbors, and Euclidean distance is not a good measurement for translated data. Euclidean distance, Hausdorff distance and SSP distance are discussed, and SSP distance is used for improved LLE algorithm. Two methods are put forward for improving LLE algorithm. One is aligning input data of original LLE algorithm, the other is modifying LLE algorithm itself. SSP distance is used to find neighbors, translated data based on SSP distance is used for gaining reconstruction weight, and the plot for each dimension of LLE representation is used for visualization of LLE representation. Motion analysis experiments results show, improved LLE algorithm is better than original LLE algorithm for translated data, and obtains better visualization of LLE representation.

Keywords: LLE, NDR, Motion analysis.

1 Introduction

Data analysis and visualization are very important for many areas of science and technology [1]. Finding compact representations of high-dimensional data is the fundamental problem of dimensionality reduction. The motivation of dimensionality reduction includes: reducing storage requirements, eliminating noise, extracting feature for recognition and projecting data to a low-dimensional space. Human brain representations the world through dealing with data form large numbers of sensory inputs. Coherent structure in the world leads to strong correlations between inputs, which will result in observations that lie on a smooth low-dimensional manifold. If the data can be represented as points in a high-dimensional vector space, it should inherently have a much more compact representation.

* This paper is sponsored by the project (No. 04KJB510167) from the education department of Jiangsu, China.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 733–742, 2007.

Dimensionality reduction method can be divided into two kinds: linear dimensionality reduction methods and nonlinear dimensionality reduction (NDR) methods. Linear dimensionality reduction methods include: PCA (principal component analysis), ICA (independent component analysis), LDA (linear discriminate analysis), LFA (local feature analysis), and so on. Nonlinear dimensionality reduction methods also can be categorized into two kinds: kernel-based methods and eigenvalue-based methods. Kernel-based methods include: KPCA (kernel principal component analysis), KICA (kernel independent component analysis), KDA (kernel discriminate analysis), and so on. Eigenvalue-based methods include: LLE（locally linear embedding）[1], ISOMAP[2], Laplacian Eigenmap[3], and so on.

LLE is a fast NDR algorithm, which finds local geometry in high dimension space and generates a projection to lower dimensional space that preserves original local geometry. LLE algorithm is inherent translation, scale and rotation invariant. The implementation of LLE is simple and is based on standard linear algebra methods. The coordinates of low-dimensional LLE representation spaces usually have meaningful attributes.

LLE is perfect in theory and can obtain good low-dimensional representation of man-made data and some practical data. LLE can't always give good low-dimensional representation of practical data. Translation of data is usually appeared in practical data. The form of data translation depends on the type of data. For example, the translation of speech signal is lies on a horizontal line, and the translation of image signal is lies on an image plane. LLE uses Euclidean distance to find neighbors, and Euclidean distance is not a good measurement for translated data. So an improved LLE algorithm is needed. There will be two methods for improving LLE algorithm. One is aligning properly input data of original LLE algorithm to eliminate the influence of data translation, the other is modifying LLE algorithm itself.

The rest part of this paper is arranged as follows. LLE algorithm is introduced briefly in section 2. Similarity measurement will be discussed in section 3. Improved LLE algorithm is depicted in section 4. Experiments and results are in section 5. Section 6 is conclusion.

2 LLE Algorithm

The input is matrix $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i \in R^D$. The output is matrix $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N\}$, where $\boldsymbol{y}_i \in R^d$ and $d << D$. For each vector $\boldsymbol{y}_i$, repeat following three steps:

(1) Find K nearest neighbors $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{iK}\}$;

(2) Find weight matrix $\boldsymbol{W} = \{W_{ij} \mid i = 1,2,...,N; j = 1,2,...,K\}$, which minimizes following cost function,

$$\varepsilon(\boldsymbol{W}) = \sum_{i=1}^{N} \left| \boldsymbol{x}_i - \sum_{j=1}^{K} W_{ij} \boldsymbol{x}_{ij} \right|^2 \tag{1}$$

where $\boldsymbol{W}$ also satisfies conditions: $\sum_{j=1}^{K} W_{ij} = 1$ and $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a neighbor of $\boldsymbol{x}_i$;

(3) Find d dimension embedding vector $\boldsymbol{y}_i$, which minimizes following cost function,

$$\Phi(\boldsymbol{Y}) = \sum_{i=1}^{N} \left| \boldsymbol{y}_i - \sum_{j=1}^{K} W_{ij} \boldsymbol{y}_j \right|^2 \tag{2}$$

3 Similarity Measurement

Usually Euclidean distance or normalized dot products are used for finding K nearest neighbors in LLE algorithm [1]. Translation of data is one of the commonest phenomena, and Euclidean distance is unsuitable for translated data.

3.1 Euclidean Distance

The definition of Euclidean distance between two vector $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is

$$D_E = \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_2 = \sqrt{\sum_{n=1}^{D} \left(\boldsymbol{x}_i(n) - \boldsymbol{x}_j(n) \right)^2} \tag{3}$$

Figure 1(a) is the original silhouette of gait image. Figure 1(b) is the horizontal translation versions of figure 1(a). Figure 1(c) is the Euclidean distance between original image and its horizontal translation versions. Figure 1 shows, directed Euclidean distance between two images is not a good distance measurement.

Figure 2(a) is another original silhouette of gait image for same person. Figure 2(b) is the horizontal translation versions of figure 2(a). Figure 2(c) is the Euclidean distance between horizontal translation versions of figure 2(a) and figure 1(a). Figure 2(c) shows, the minimum Euclidean distance is obtained when the horizontal shift is 1 pixel. Figure 2 also shows, directed Euclidean distance between two images is not a good distance measurement.

3.2 Hausdorff Distance

Hausdorff distance is a kind of distance measurement between two point sets [4]. Let $\boldsymbol{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_m\}$ and $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n\}$ denote two finite point sets. The definition of Hausdorff distance is

$$H(\boldsymbol{A},\boldsymbol{B}) = \max(h(\boldsymbol{A},\boldsymbol{B}), h(\boldsymbol{B},\boldsymbol{A})) \tag{4}$$

$h(A,B)$ is directed Hausdorff distance, and it is defined as

$$h(\boldsymbol{A},\boldsymbol{B}) = \max_{\boldsymbol{a}\in\boldsymbol{A}} \min_{\boldsymbol{b}\in\boldsymbol{B}} \|\boldsymbol{a}-\boldsymbol{b}\| \tag{5}$$

The modified Hausdorff distance is defined as

$$h_{\text{mod}}(\boldsymbol{A},\boldsymbol{B}) = \frac{1}{|\boldsymbol{A}|} \sum_{\boldsymbol{a}\in\boldsymbol{A}} \min_{\boldsymbol{b}\in\boldsymbol{B}} \|\boldsymbol{a}-\boldsymbol{b}\| \tag{6}$$

Modified Hausdorff distance using the average of single point distances, so is robust for outliers. Hausdorff distance is useful for shape matching. Hausdorff distance eliminates the influence of point translation to some extent.

3.3 SSP Distance

BenAbdelkader directly models human motion, and believes the dynamic feature of gait is encoded in pairwise image similarities of gait images, and gives the definition of self-similarity plot (SSP) [5]. The definition of SSP is

$$S(t_1,t_2) = \min_{|dx,dy|<r} \sum_{(x,y)\in B_{t_1}} \left| \boldsymbol{O}_{t_1}(x+dx, y+dy) - \boldsymbol{O}_{t_2}(x,y) \right| \tag{7}$$

where, t_1 and t_2 are numbers of silhouette images, B_{t_1} is the bounding box of silhouette image t_1, r is a small search radius, and $\boldsymbol{O}_{t_1}$ and $\boldsymbol{O}_{t_2}$ are silhouette images. When SSP is used for gait alignment, we let $t_2 = 1$ and $\boldsymbol{f}_{t_1}$ is aligned with dx and dy.

The definition of SSP distance between two vector $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is

$$D_S(\boldsymbol{x}_i, \boldsymbol{x}_j) = \min(D_E(\boldsymbol{x}_i, T(\boldsymbol{x}_j))) \tag{8}$$

where $T(\boldsymbol{x}_j)$ is the translation version of $\boldsymbol{x}_j$. The operator T is depends on the type of data. SSP distance removes the influence of data translation to a great extent.

4 Improved LLE Algorithm

Two methods can be used for improving LLE algorithm. One is aligning input data of original LLE algorithm, the other is modifying original LLE algorithm itself.

4.1 Aligning Input Data of LLE Algorithm

For linear or nonlinear dimensionality reduction methods, data alignment is very important step. The input of LLE algorithm is matrix $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i \in R^D$. The alignment version of $\boldsymbol{X}$ is

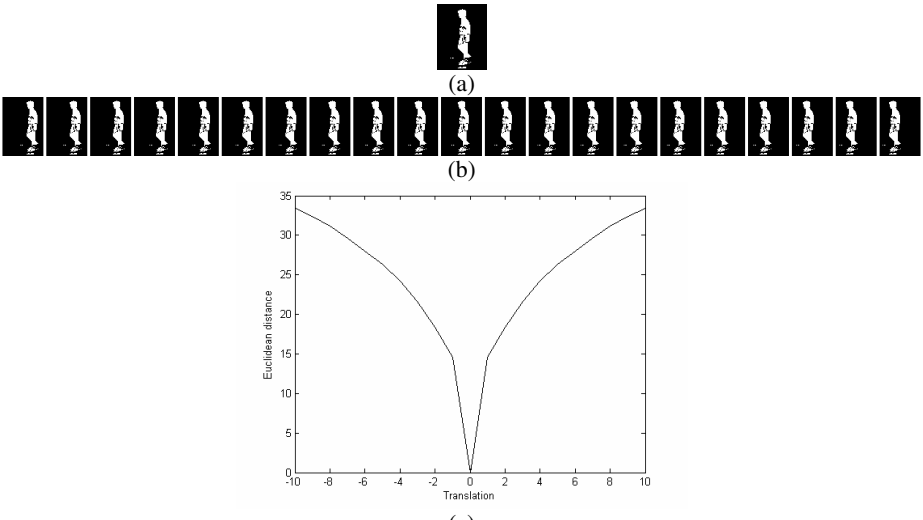


Fig. 1. (a) Original image, (b) Horizontal translation version of original image, (c) Euclidean distance between original image and its horizontal translation versions

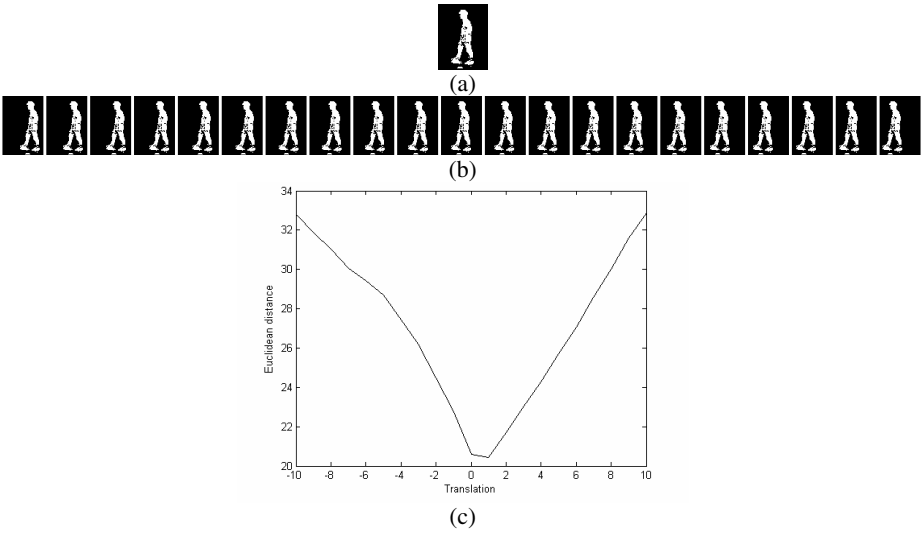


Fig. 2. (a) Original image, (b) Horizontal translation version of original image, (c) Euclidean distance between original image figure 1(a) and horizontal translation versions of original image figure 2(a)

$$T_S(\boldsymbol{X}) = \{T_S(\boldsymbol{x}_1), T_S(\boldsymbol{x}_2), ..., T_S(\boldsymbol{x}_N)\} \tag{9}$$

$T_S(\boldsymbol{x}_i)$ satisfies

$$D_E(T_S(\boldsymbol{x}_i), \boldsymbol{x}_R) = D_S(\boldsymbol{x}_i, \boldsymbol{x}_R) \tag{10}$$

$\boldsymbol{x}_R$ is a reference vector, and it may be $\boldsymbol{x}_R = \boldsymbol{x}_1$. A more sophisticated $\boldsymbol{x}_R$ is needed.

4.2 Improved LLE Algorithm

The input is matrix $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i \in R^D$. The output is matrix $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N\}$, where $\boldsymbol{y}_i \in R^d$ and $d << D$. For each vector $\boldsymbol{x}_i$, repeat following three steps:

(1) Using SSP distance to find K nearest neighbors $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{iK}\}$;

(2) Find weight matrix $\boldsymbol{W} = \{W_{ij} \mid i = 1,2,...,N; j = 1,2,...,K\}$, which minimizes following cost function,

$$\varepsilon(\boldsymbol{W}) = \sum_{i=1}^{N} \left| \boldsymbol{x}_i - \sum_{j=1}^{K} W_{ij} T_S(\boldsymbol{x}_{ij}) \right|^2 \tag{11}$$

$T_S(\boldsymbol{x}_{ij})$ is the translation version of $\boldsymbol{x}_{ij}$, which satisfies

$$D_E(\boldsymbol{x}_i, T_S(\boldsymbol{x}_{ij})) = D_S(\boldsymbol{x}_i, \boldsymbol{x}_{ij}) \tag{12}$$

$\boldsymbol{W}$ also satisfies conditions: $\sum_{j=1}^{K} W_{ij} = 1$ and $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a neighbor of $\boldsymbol{x}_i$;

(3) Find d dimension embedding vector $\boldsymbol{y}_i$, which minimizes following cost function,

$$\Phi(\boldsymbol{Y}) = \sum_{i=1}^{N} \left| \boldsymbol{y}_i - \sum_{j=1}^{K} W_{ij} \boldsymbol{y}_j \right|^2 \tag{13}$$

(4) Visualization of lower-dimensional embedding vector $\boldsymbol{y}_i$. Classical method for visualization of lower-dimensional embedding is the plot of 1D, 2D or 3D LLE

representations. Another available method is the plot for each dimension of lower-dimensional LLE representation.

5 Motion Analysis Experiments and Results

Gait images come from CMU MOBO database [6]. CMU MOBO database has 25 persons, 6 visual angles and 4 kinds of walk: slow walk, fast walk, slow incline walk and slow walk with a ball.

5.1 Comparison Between Improved and Original LLE Algorithm

Figure 3(a) is the 1D LLE representation of gait sequence using original LLE algorithm. Figure 3(b) is the 1D LLE representation of gait sequence using original LLE algorithm and input data alignment. Figure 3(c) is the 1D LLE representation of gait sequence using improved LLE algorithm. Figure 3 shows, three methods have similar good 1D LLE representation.

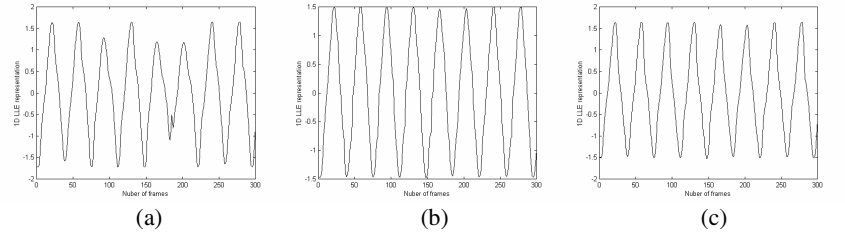
Figure 4(a) is the 1D LLE representation of gait sequence using original LLE algorithm. Figure 4(b) is the 1D LLE representation of gait sequence using original LLE algorithm and input data alignment. Figure 4(c) is the 1D LLE representation of gait sequence using improved LLE algorithm. Figure 4 shows, original LLE algorithm can not give good 1D LLE representation, and input data alignment or improved LLE algorithm has similar good 1D LLE representation.

Figure 5(a) is the 1D LLE representation of gait sequence using original LLE algorithm. Figure 5(b) is the 1D LLE representation of gait sequence using original LLE algorithm and input data alignment. Figure 5(c) is the 1D LLE representation of gait sequence using improved LLE algorithm. Figure 5 shows, only improved LLE algorithm can have good 1D LLE representation.

Experiments results show, input data alignment based LLE algorithm is better than original LLE algorithm, and improved LLE algorithm is better than input data alignment based LLE algorithm.

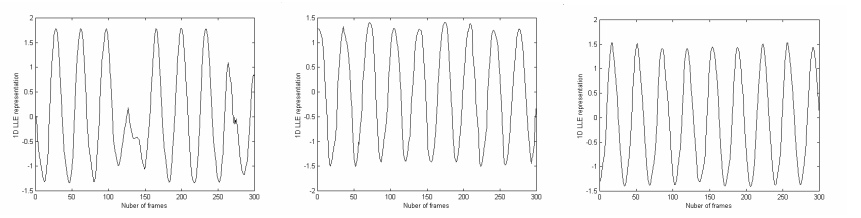
5.2 Visualization of Low-Dimensional LLE Representation

Figure 6(a) is the plot for 1D LLE representation of gait sequence using improved LLE algorithm, figure 6(b) is the plot for 2D LLE representation of gait sequence using improved LLE algorithm, and figure 6(c) is the plot for 3D LLE representation of gait sequence using improved LLE algorithm.



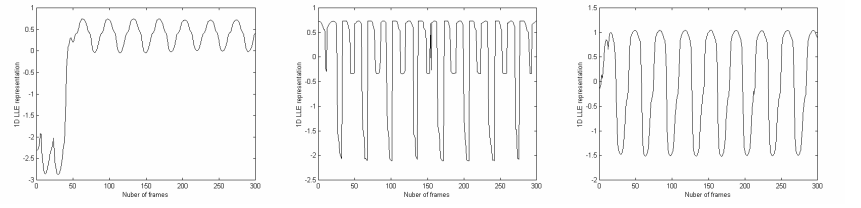
(a) (b) (c)

Fig. 3. 1D LLE representation of gait sequence using original LLE algorithm, input data alignment and improved LLE algorithm



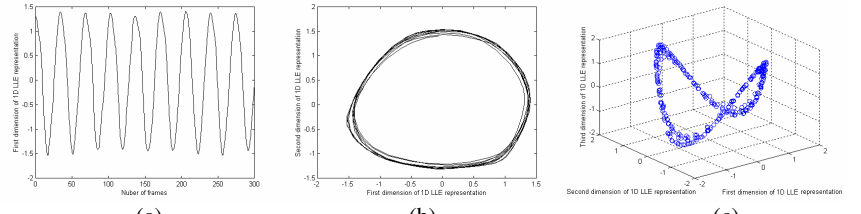
(a) (b) (c)

Fig. 4. 1D LLE representation of gait sequence using original LLE algorithm, input data alignment and improved LLE algorithm



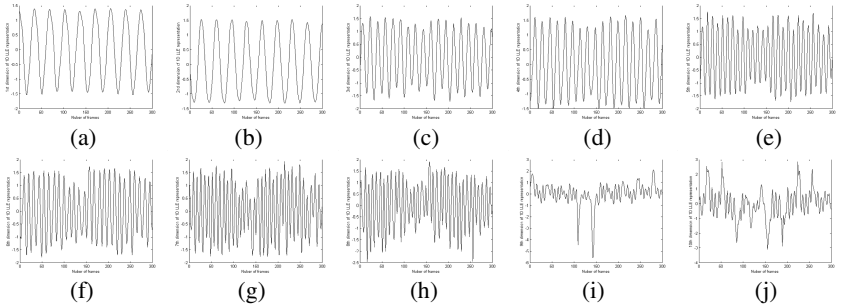
(a) (b) (c)

Fig. 5. 1D LLE representation of gait sequence using original LLE algorithm, input data alignment and improved LLE algorithm



(a) (b) (c)

Fig. 6. 1D, 2D and 3D LLE representation of gait sequence using improved LLE algorithm



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Fig. 7. 1st~10th dimension of LLE representation for gait sequence using improved LLE algorithm

Figure 6(a) shows, Zero-crossing, local maximum and local minimum of 1D LLE representation represents gait cycle, and the shape of 1D LLE representation represents space extension of gait [7]. Figure 6(b) and figure 6(c) show, 2D and 3D LLE representation also represent gait cycle and dynamic features of gait.

Figure 7(a) is the first dimension of LLE representation using improved LLE algorithm, figure 7(b) is the second dimension of LLE representation using improved LLE algorithm, and so on.

Figure 7 shows, different dimension of LLE representation represents different dynamic feature of gait in different scale. The smaller dimension of LLE representation shows dynamic features of gait in lager scale, the larger dimension of LLE representation shows dynamic features of gait in smaller scale.

6 Conclusions

Improved LLE algorithm is provided. Generally there is shift in practical data. The form of shift depends on the type of data. Data alignment and registration is a very important step for linear or nonlinear dimensionality reduction methods. Original LLE algorithm uses Euclidean distance to find neighbors, and Euclidean distance is not a good measurement for translated data. Euclidean distance and Hausdorff distance are discussed, and the definition of SSP distance is given and is used for improved LLE algorithm. Two methods are brought out for improving LLE algorithm. One is aligning input data of original LLE algorithm, the other is modifying LLE algorithm itself. In improved LLE algorithm, SSP distance is used to find neighbors, translated data based on SSP distance is used for obtaining reconstruction weight, and the plot for each dimension of LLE representation is used for visualization of LLE representation. Motion analysis experiments results show, improved LLE algorithm is better than original LLE algorithm for translation data and gains better visualization of LLE representation.

In further work, improved LLE algorithm will be used for gait analysis and recognition, gesture analysis and recognition, satellite cloud image analysis and recognition, video analysis, and so on. Because ISOMAP algorithm is still not robust for data translation just as LLE algorithm, improved ISOMAP will be studied in further work.

Acknowledgement

I would like to thank Ralph Gross for warm-hearted help of CMU MOBO database download from their web site and excellent suggestions of gait recognition, especially view dependent and independent of gait cycle detection.

References

1. Roweis, S.T.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290, 2323–2326 (2000)
2. Tenebaum, J.B.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290, 2319–2323 (2000)

3. Belkin, M., Niyogi, P.: Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. Neural Information Processing Systems 14, 585–591 (2002)
4. Jesorsky, O., Kirchberg, K.J., Frischholz, R.W.: Robust Face Detection Using the Hausdorff Distance. In: Bigun, J., Smeraldi, F. (eds.) AVBPA 2001. LNCS, vol. 2091, pp. 90–95. Springer, Heidelberg (2001)
5. BenAbdelkader, C., Cutler, R., Davis, L.: Motion-based Recognition of People in EigenGait Space. In: Proc. of the Fifth IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp. 254–259. IEEE Press, New York (2002)
6. Gross, R., Shi, J.: The CMU Motion of Body (MoBo) Database. Technical Report, CMU-RI-TR-01-18, Robotics Institute, Carnegie Mellon University (2001)
7. Li, H., Li, X.: LLE Based Gait Analysis and Recognition. In: Li, S.Z., Lai, J.-H., Tan, T., Feng, G.-C., Wang, Y. (eds.) SINOBIOMETRICS 2004. LNCS, vol. 3338, pp. 671–679. Springer, Heidelberg (2004)

Hybrid GA Based Online Support Vector Machine Model for Short-Term Traffic Flow Forecasting

Haowei Su and Shu Yu

College of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, P.R. China
gzsuhw@tom.com, yushu_scut@126.com

Abstract. In this paper, a hybrid genetic algorithm (GA) based online support vector machine (OSVM) prediction model for short-term traffic flow forecasting is proposed, according to the data collected sequentially by the probe vehicle or the loop detectors, which can update the forecasting function in real time via online learning way, and the parameters used in the OSVM were optimized by GA. As a result, it is fitter for the real engineering application. The GA based OSVM model was tested by using the I-880 database, the result shows that this model is superior to the back-propagation neural network (BPNN) model.

1 Introduction

Intelligent Transportation System (ITS) emerges as the times requires. The basic idea of the ITS is: on the precondition that the current road situation is not varied, and recur to the high precision and real time predictive method of traffic flow [1-4], so as to achieve effective control of traffic and transport guide from the largest extent to ease the problem of traffic jams.

According to the theory, the traffic flow forecasting research can be divided into two types. One type is determined based on the mathematical model [5], but the short-term traffic flow prediction is more influenced by the stochastic interferential factors than the long-term one, the uncertainty is greater and the disciplinarian laws are less obvious. Thus using the short-term traffic prediction models based on the classical mathematical methods, the precision of forecast can not satisfactorily meet the demand of real-time traffic control and guidance in ITS.

The second type is knowledge-based intelligent model of forecasting methods [6], in which the typical representative one is the BPNN model. As BPNN learning algorithm used gradient descent algorithm and weights regulation to minimize the objective function, the objective function was set by the square sum of the margin between the input and output values, so the BPNN led to excessively emphasize the learning mistakes and the over fitting problem appeared inevitably.

Based on the fact that the traffic flow prediction is equivalent to the function estimates and approximation [7], it can be handled as the function estimates issue. Support vector machine (SVM) is a new type of learning machine [8], using structured risk minimization (SRM) principles to perform regression or pattern recognition. SVM

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 743–752, 2007.

can solve some flaws of the neural networks, and has many unique advantages in the fields of small samples and high-dimensional nonlinear manifested [9-10].

Currently, the traffic flow forecast based on support vector machine is a new and hot research field, but still at a preliminary stage. In 2004, a traffic forecasting method was proposed by Wang Jisheng et al [11], which based on support vector machines theory. The model was solved via the LIBSVM algorithm and the results were better than the BPNN. Zhang Chaoyuan et al [12] proposed a least square support vector machine version model to forecast the traffic flow time series, and the algorithm based on the LS-SVM was presented.

The traffic flow prediction has a major characteristic of real-time nature. On the basis of research on support vector machine learning, Yin Ying et al [13] designed a least square support vector machine simulation and real-time traffic flow forecasting system using Matlab language, which gave a new way for visualization expression of the traffic flow guidance data. A real-time traffic flow prediction model based on support vector machine was given by Xu Qihua and Yang [14]. Through sequential minimal optimization (SMO) algorithm, this model can effectively forecast for noise traffic data. In the literature [15], after the analysis of the characteristics of urban traffic flow, the authors introduced the kernel machine methods, and compared performance of the compound and conventional kernels was also shown. A short-term traffic flow forecasting model based on support vector machine was proposed by Yang Zhaosheng et al [16] in 2006, on the basis of concluding variety of traffic flow forecasting models and the mature thought about the nonlinear, complexity and uncertainty of traffic system. Compared with the BPNN model, the results showed that in the fields of accuracy, convergence time, generalization and optimality, SVM-based model is superior to neural network-based models. SVM-based forecasting method was applied to the field of traffic incident detection [17], and had shown good results.

The parameters selected in the SVM are much important to the efficiency of the prediction model, especially in the real application. In order to gain the optimal parameters, a global optimize process should be implement.

Genetic algorithm (GA) [18] is a global stochastic algorithm, which derives from the ideas of selection process in nature and genetic mechanism. GAs were successfully applied to many fields [19], because of their superiority of implementation on massively parallel architecture, compared with traditional optimization methods in searching for the global optimum of complex problem. However, it is not easy to regulate GA's convergence so that GA often suffers from premature convergence [20].

By contrast with GA, simulated annealing (SA) [21] algorithm employs certain probability to escape from local optima and the search process can be controlled by the cooling schedule. Since the complementary strengths of GA and SA, how to integrate GA with SA to achieve more efficient optimization results was widely studied in both theory and application areas [22].

Therefore, in this paper, an online learning approach using support vector machine model was proposed, which can dynamically update the forecasting function via the data collected continuously by the probe vehicle or loop detectors. And the parameters used in the OSVM were determined by the hybrid GA. The goal is to develop a rapid, real-time, and optimal short-term traffic flow prediction model with high generalization ability.

2 Hybrid GA Based Online Support Vector Machine

2.1 Least Square Support Vector Machine

In 1999, Suykens [23] presented the least square_support vector machine algorithm, in which the main idea is to introduce the least square system into the standard SVM.

Assuming study set is

$$\boldsymbol{S}=\left\{s_i \mid s_i=(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in R^n, y_i \in R, i=1,2,\cdots,l\right\},$$

where the regression function expressed as:

$$y(\boldsymbol{x})=\boldsymbol{w}\cdot\varphi(\boldsymbol{x})+b \tag{1}$$

The LS-SVR proposed by Suykens et al is to solve the following problem:

$$\begin{cases}\min Q(\boldsymbol{w},\boldsymbol{e})=\dfrac{1}{2}\|\boldsymbol{w}\|^2+\dfrac{\gamma}{2}\sum\limits_{i=1}^{l}e_i^2 \\ s.t.\ y_i=\boldsymbol{w}\cdot\varphi(\boldsymbol{x}_i)+b+e_i, i=1,2,\cdots,l\end{cases} \tag{2}$$

It can be known from formula (2), that the balance equation is expressed as:

$$\begin{bmatrix}0 & \vec{\boldsymbol{I}}^T \\ \vec{\boldsymbol{I}} & \boldsymbol{Z}\boldsymbol{Z}^T+\gamma^{-1}\boldsymbol{I}\end{bmatrix}\begin{bmatrix}b \\ \boldsymbol{\alpha}\end{bmatrix}=\begin{bmatrix}0 \\ \boldsymbol{y}\end{bmatrix} \tag{3}$$

where $\boldsymbol{Z}=[\varphi(\boldsymbol{x}_1),\varphi(\boldsymbol{x}_2),\cdots,\varphi(\boldsymbol{x}_l)]^T$, $\boldsymbol{y}=[y_1,y_2,\cdots,y_l]^T$, $\vec{\boldsymbol{I}}=[1,1,\cdots,1]^T$, $\boldsymbol{e}=[e_1,e_2,\cdots,e_l]^T$, $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\cdots,\alpha_l]^T$.

From the Mercer condition:

$$\varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}_j)=k(\boldsymbol{x}_i,\boldsymbol{x}_j)\equiv\Omega_{ij},\ i,j=1,2,\cdots,l, \tag{4}$$

here $k(\cdot,\cdot)$ is a kernel function, which is often used as Gauss kernel

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2/(2\sigma^2))$$

$\boldsymbol{\Omega}+\gamma^{-1}\boldsymbol{I}$ is called as kernel correlation matrix, let $\boldsymbol{A}\equiv\boldsymbol{\Omega}+\gamma^{-1}\boldsymbol{I}$, then formula (4) can be written as:

$$\begin{bmatrix}0 & \vec{\boldsymbol{I}}^T \\ \vec{\boldsymbol{I}} & \boldsymbol{A}\end{bmatrix}\begin{bmatrix}b \\ \boldsymbol{\alpha}\end{bmatrix}=\begin{bmatrix}0 \\ \boldsymbol{y}\end{bmatrix} \tag{5}$$

the regression function in formula (1) is:

$$y(\boldsymbol{x})=\boldsymbol{w}\cdot\varphi(\boldsymbol{x})+b=\sum_{i=1}^{l}\alpha_i\varphi(\boldsymbol{x}_i)\varphi(\boldsymbol{x})+b=\sum_{i=1}^{l}\alpha_i k(\boldsymbol{x}_i,\boldsymbol{x})+b \tag{6}$$

$\boldsymbol{\alpha}, b$ are called the regression parameters.

It can be seen from formula (6) that the key to get the regression parameters is the computation of the matrix, $\boldsymbol{A}^{-1}$.

2.2 Online SVM Learning Algorithm

Conducting short-term traffic flow forecasting, the system continuously collects the data on the flow and return to the prediction model in sequence, after that the model should be adaptively adjusted according to the new collected sample.

When new samples (x_{l+1}, y_{l+1}) were added to the learning set, the kernel correlation matrix was changed into:

$$A_{l+1} = \begin{bmatrix} A_l & B^T \\ B & c \end{bmatrix} \tag{7}$$

where A_l, A_{l+1} were kernel correlation matrix of learning set S and $S \bigcup \{ s_{l+1} \}$

$$B = \begin{bmatrix} y_{N+1} & \Omega_{N+1,1} & \Omega_{N+1,2} & \cdots & \Omega_{N+1,N} \end{bmatrix}$$

If A_l^{-1} can be used to obtain A_{l+1}^{-1} without the totally recalculating, then the online SVM learning task is done. In fact, this is the improvement to the online LS-SVR algorithm which was the LS-SVR based research result of liu et al [20]. In paper [21] there can be shown:

$$A_{l+1}^{-1} = \begin{bmatrix} A_l & B^T \\ B & c \end{bmatrix}^{-1} = \begin{bmatrix} \left[A_l - \frac{1}{c} B^T B \right]^{-1} & A_l^{-1} B^T \left[B A_l^{-1} B^T - c \right]^{-1} \\ \left[B A_l^{-1} B^T - c \right]^{-1} B A_l^{-1} & \left[c - B A_l^{-1} B^T \right]^{-1} \end{bmatrix} \tag{8}$$

$$\left[A_l - \frac{1}{c} B^T B \right]^{-1} = A_l^{-1} - A_l^{-1} B^T \left[-c + B A_l^{-1} B^T \right]^{-1} B A_l^{-1} \tag{9}$$

From (8-9), on the basis of the original results, the prediction function should update as following, according to the new samples added:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l+1} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{10}$$

2.3 Hybrid GA

The hybrid GA was used to optimal the parameters γ and σ in the OSVM, the object value is the training accuracy.

GA initializes a population. The population evolves from parent generation to child generation by three basic operators: the selection, the crossover and the mutation. Then the parent and child populations are united as an extended population. SA is used as a Bolzmann reduction operator to reduce the extended population to original size, so a reduced population comes into being. To control the population diversity, the diversity function is used to quantify the degree of parent and reduced population diversity, and in terms of which the one with superior diversity must have a higher probability to be chosen as the new population. The structure of the presented algorithm can be seen in the follows:

Begin
While not stop **do**
Initialize Population (P_0)
$k = 0$
While not stop **do**
Do n/2 times
{
Select two parents from P_k
Generate two children by using crossover
Mutate the two children
Introduce the children in the child population CH
}
Make the extended population $P_k \cup CH$
Introduce the n/10 best individuals of the extended population in the elitist pool
Make the reduced population $P^{"}$ by reduce the extended population to the original size
Choose P_k or $P^{"}$ to be P_{k+1} in terms of their degree of population diversity
Modify temperature (c_k)
$k = k + 1$
End

End

End

Both the variable and object function are continuous. In order to improve the accuracy and the convergence speed of GA, we use real coding. So one representation of the solution of an individual can be $X = [x_1, x_2] = [\gamma, \sigma]$.

Selection: the selection operator used is "roulette wheel selection". The probability of reproduction for the individual i is given by: $P(i) = F(i) / \sum_{i \in I} F(i)$.

Crossover: the crossover operator used is convex crossover.

$$\begin{aligned} X_1' &= \lambda_1 X_1 + \lambda_2 X_2 \\ X_2' &= \lambda_2 X_1 + \lambda_1 X_2 \end{aligned}, \lambda_1 + \lambda_2 = 1, \quad \lambda_1 > 0, \lambda_1 > 0$$

Mutation: The mutation operator used is as followed: $X' = X + r \cdot d$, d is the approximately grads, the i th variable of d can be computed by

$$d_i = \frac{f(x_1, \cdots x_i + \Delta x_i, \cdots, x_n)}{\Delta x_i}$$

r is a non- negative random real number.

The reduction operator consists of sampling a Boltzmann probability distribution in the extended population (the union of the parent and child populations). The value of this probability distribution depends on the fitness function of the individuals in the population. If the size of the population is n, in the case of maximization, it works as follows:

While not n individuals have been selected **do**
{
Choose randomly an individual i from the extended population
If $F(i) > \overline{F}$ then
Select individual i for the reduced population
Else
Select individual i with probability equal to $\exp(\frac{F(i)-\overline{F}}{c_k})$
}
End

Where $F(i)$ is the fitness value of the ith individual and $\overline{F}$ is the mean value of fitness function F in the parent population.

3 Hybrid GA Based OSVM Model for Short-Term Traffic Flow Forecasting

The probe vehicle or loop detectors can be used to collect the traffic information [24]. Subject to random factors (e.g. transmission errors, etc.), it can not be avoided to lose data accuracy such as data errors and data loss, so the data preprocessing need to be implemented to correction errors, of which the threshold test and traffic flow theory-based check are two commonly used methods. At last, the data should be normalized treatment to improve the efficiency of computation.

The short-term traffic flow forecasting process can be described as followed:

Step1: Chose N traffic flow samples as initial training set

$$\boldsymbol{S} = \{s_i \mid s_i = (\boldsymbol{x}_i, y_i), i = 1,2,\cdots,N\},$$

According to formula (1)-(6) of LS-SVM algorithm, the initial prediction regression function is obtained. Then the prediction value for traffic flow y_j is:

$$y_j = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \tag{11}$$

Step2: Alone with the data collected continuously, the prediction regression function should be updated in time. Assuming a new samples are chosen to add in the training set, so

$$\boldsymbol{S} = \{s_i \mid s_i = (\boldsymbol{x}_i, y_i), i = 1,2,\cdots,N,N+1\}$$

According to formula (7)-(10) of OSVM algorithm, the regression function can be reconstructed, the prediction value for traffic flow y_j is:

$$y_j = \sum_{i=1}^{N+1} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \tag{12}$$

The Guass kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{exp}(-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 / (2\sigma^2))$$

was used as the kernel function.

Step3: Use hybrid GA to optimal the parameters of the online support vector machine.

4 Model Validation and Comparison

In order to analyze the forecast results better, five error indicators are introduced:

Relative Error:

$$rerr = \frac{v_{pred}(t) - v_{real}(t)}{v_{real}(t)},$$

Mean Relative Error:

$$mrerr = \frac{1}{N}\sum_t \frac{v_{pred}(t) - v_{real}(t)}{v_{real}(t)},$$

Mean Absolute Relative Error:

$$marerr = \frac{1}{N}\sum_t \frac{\left|v_{pred}(t) - v_{real}(t)\right|}{v_{real}(t)},$$

Root Mean Square Relative Error:

$$rmrerr = \sqrt{\frac{1}{N}\sum_t \left(\frac{v_{pred}(t) - v_{real}(t)}{v_{real}(t)}\right)^2},$$

Equalization Coefficient (EC):

$$EC = 1 - \frac{\sqrt{\sum_t \left(v_{pred}(t) - v_{real}(t)\right)^2}}{\sqrt{\sum_t \left(v_{pred}(t)\right)^2} + \sqrt{\sum_t \left(v_{real}(t)\right)^2}}$$

EC means the difference between the predicted and actual fit in a well fitting, when it is above 0.90.

In order to compare the prediction results of SVM based model with that of popular neural network based method, we construct the BPNN model for the traffic flow forecast. The BPNN-based traffic flow prediction model is composed of data processor, input layer, output layer and hidden layer. It can be seen in the Fig.1.:

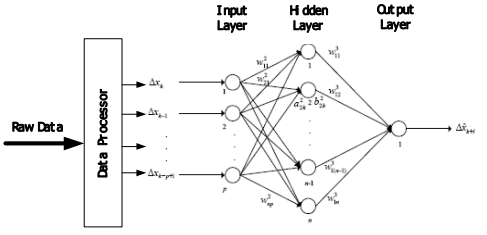


Fig. 1. BPNN-based traffic flow prediction model

The formula of the prediction model is as following:

$$Q_t = \left[\frac{exp(-\sum_{j=1}^{i} v_{jt})}{1+exp(-\sum_{i=1}^{n} w_{ij} x_i \theta_j)} + \gamma_i \right]^{-1} \tag{13}$$

here, Q_t is the forecast value, w_{ij} is the weight of connection between input layer and hidden layer, θ_j is the threshold of hidden layer unit, v_{jt} is the weight of connection between hidden layer and output layer, γ_i is the threshold of output layer unit.

To validate the correctness and accuracy of the forecasts model, the I-880 database [25] was used to test. Since there were large amount of raw data in this database, we simply randomly selected some samples in a short period in each day, and the data were regulated for computational convenience. The BPNN-based short-term traffic flow prediction model was programmed by using Matlab7.0.1 neural network toolbox, the hybrid GA based OSVM forecast model was programmed by using Microsoft Visual C++ 6.0 complier. The operating environment is: CPU Pentium 1.5 MHz, Memory 1G, Microsoft Windows XP operation system.

Table 1. Results of two models

Model	Evaluate Indicators			
	mrerr%	marerr%	rmrerr%	EC
OSVM	0.51	5.25	6.84	0.97
BPNN	0.77	1.46	13.28	0.933

According to the results of Table 1, the hybrid GA based OSVM model gave higher accuracy than the BPNN-based model, and reached a high value of EC fitting. Besides the OSVM updated forecast function via online learning algorithm, which was more suited for the practical application. In generalized performance, the OSVM model is also superior to the BPNN-based model. The basis of OSVM was support vector machines, so the forecast error is relatively stable even the fitting error was large. But neural network appear over fitting easily, when fitting errors are reduced the forecast error will get larger soon. Moreover, the training of OSVM in fact is a

convex quadratic programming problem, which can obtain the global optimal solution with hybrid GA optimized parameters. Contrarily, gradient descent algorithm was used for neural network, which often result in local optimal solution.

5 Conclusions

The performance of many components in intelligent transportation systems depends heavily on the quality of short-term traffic forecasts. Considering the real-time, nonlinear, complexity and uncertainty in traffic problems, a new hybrid GA based online support vector machine was proposed for designing short-term traffic flow prediction model. This method was based on least square support vector machine, and used online learning strategy to dynamic update forecast function, which was more suited for the practical application. Moreover, the designed online learning algorithm used hybrid GA to optimal the parameters. The I-880 database was used to test the hybrid GA based OSVM model and the BPNN-based model, and the results shown that the hybrid GA based OSVM was superior to the BPNN work in accuracy, generalized ability and optimization. The further work is to find an effective strategy to dump the useless history data in the training set, in order to avoid reducing the convergence speed when the learning samples increase in number.

References

1. Zhu, Z., Yang, Z.S.: Real time traffic flow prediction model based on ANN. China Journal of Highway and Transport 11(4), 89–92 (1998)
2. Yang, Z.S., Gu, Y.: Real time and dynamic traffic flow forecast research. Journal of Highway and Transportation Research and Development 15(3), 4–7 (1998)
3. Yang, Z.S.: Theories and models of urban traffic guidance system. China Communications Press, Beijing (2000)
4. Li, C.J., Yang, R.G., Zhang, J.S.: Traffic flow forecasts based on wavelet analysis. Journal of Computer Applications 23(12), 7–8 (2003)
5. He, G.J., Li, Y., Ma, S.F.: Short-term traffic flow prediction based on mathematic models. Systems engineering-theory & practice 12, 51–56 (2000)
6. He, G.Q.: Introduction to ITS systems engineering. China railway publishing house, Beijing (2004)
7. Yang, Z.S., Jiang, G.Y.: Theories and models of urban traffic guidance system based on high-order neural networks. Journal of Highway and Transportation Research and Development 6, 16–19 (1998)
8. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
9. Cheng, H., Cheng, L.H., et al.: The Appliance of BP-Network and SVM in Approach of Non-linear Function. Aeronautical Computer Technique 34(3), 27–30 (2004)
10. Xu, Q.H., Shi, J.: Aero-Engine Fault Diagnosis Based on Support Vector Machine. Journal of Aerospace Power 2, 298–302 (2005)
11. Wang, J.S., Gao, B.C., Shi, L.P.: Application of support vector machines in traffic volume forecast. Information Technology 4, 8–10 (2004)
12. Zhang, C.Y., Hu, G.H., Xu, T.Z.: Traffic flow time series prediction based on LS-SVM. Journal of Yunnan University 26, 19–22 (2004)

13. Yin, Y., Zhang, C.Y., Hu, G.H., Xu, T.Z.: Design of Real-Time Traffic Flow Simulating and Forecasting System Based on SVM. Journal of Computer Engineering and Applications 10, 197–199 (2005)
14. Xu, Q.H., Yang, R.: Traffic Flow Prediction Using Support Vector Machine Based Method. Journal of Highway and Transportation Research and Development 22(12), 131–134 (2005)
15. Jiang, G., Xiao, J.: Real-time Forecast of Urban Traffic Flow Based on Kernel Machine Method. Computer Engineering 32(17), 48–51 (2006)
16. Yang, Z.S., Wang, Y., Guan, Q.: Short-term traffic flow prediction method based on SVM. Journal of Jilin University 36(6), 881–884 (2006)
17. Qin, P.P.: Comparison of SVM and Neural Network Model for Incident Detection. Journal of Computer Engineering and Applications 34, 214–232 (2006)
18. Holland, J.H.: Adaptation in natural and artificial system. MIT Press, Cambridge, MA (1975)
19. Joine, J., Houck, C.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In: Proceedings of the First IEEE Conference on Evolutionary Compution, pp. 548–579. IEEE Press, Los Alamitos (1994)
20. Kirpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
21. Kovac, A., Glavic, P.: Retrofit of complex and energy intensive processes-I. Computers & Chemical Engineering 19, 1255–1270 (1995)
22. Kralj, A.K., Glavic, P.: Retrofit of complex and energy intensive processes. Computers & Chemical Engineering 21(Suppl.), 517–522 (1997)
23. Suykens, J.A.K., Vandewalle, J.: Least Squares Support Vector Machine Classifiers. Neural Process Letter 9, 293–300 (1999)
24. Zhang, C.B., Yang, X.G., Yan, X.P.: Traffic Data Collection System Based on Floating Cars. Computer and Communications 5(24), 31–34 (2006)
25. Petty, K.: The analysis software for the FSP project, http://ipa.eecs.berkeley.edu/ pettyk/FSP

Composed Fuzzy Rough Set and Its Applications in Fuzzy RSAR

Weigen Qiu[1,2] and Zhibin Hu[1]

[1] Computer Faculty of GuangDong University of Technology, GuangZhou, GuangDong, 510090, P.R. China
[2] State Key Laboratory of Intelligent Technology & Systems, Department of Computer Science & Technology, Tsinghua University, Beijing 100084, P.R. China

Abstract. Pawlak rough set theory is a powerful mathematic tool to deal with imprecise, uncertainty and incomplete dataset. In this paper, we study the fuzzy rough set attribute reduction (fuzzy RSAR) in fuzzy information systems. Firstly, we present the formal definition of a kind new rough set form-the composed fuzzy rough set. The second, some properties of extension forms of Pawlak rough set are also discussed. Lastly, we illustrate the fuzzy RSAR based on composed fuzzy rough set, and a simple example is given to show this approach can retain less attributes and entailing higher classification accuracy than the crisp RST-based reduction method.

Keywords: Fuzzy information system, Composed fuzzy rough sets, Fuzzy rough set, attribute reduction.

1 Introduction

It is well known many classification problems involve high-dimensional descriptions of input features. However, some existing methods tend to destroy the underlying semantics of the features after reduction or require additional information beyond the given data set. A technique that can reduce the dimensionality by using information contained within the data set and preserving the meaning of the features is clearly desirable. Rough Sets Theory (RST) can be used as such a tool to discover data dependencies and reduce the number of attributes contained in data set by purely structural methods [4].

With more than twenty years development [5], RST has indeed become an expanding research area, recent theoretical developments are collected in papers [7]. However, in traditional Pawlak RST, an equivalence relation seems to be a very stringent condition which limits its applications fields. As well known, the fuzzy set theory and rough set theory represent different aspects of uncertainty and aim to two different purpose, so many attempts have been made to combine these two theories [1,2,9]. Because the values of attributes may often be both crisp and real-valued in more and more application cases, therefore the traditional RST encounters a problem. Based on information entropy, paper [8] presents a discretization algorithm of

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, pp. 753–763, 2007.

real-valued attributes values information system for selecting cut points. Because the discretization process itself often requires some additional information beyond the aimed data sets, the paper [4] introduce a new concepts for fuzzy-rough attribute reduction based on fuzzy rough sets.

In this paper, a kind new rough set concept is presented and the composed fuzzy rough set is formally defined and its properties are discussed. The fuzzy rough attribute reduction based on the composed fuzzy rough set is illustrated with a simple example. This approach can retain less attributes and entail higher classification accuracy than the crisp RST-based reduction method. The rest of this paper is organized as follows. Section 2 discusses some related basic theory with this paper later, such as fuzzy set, Pawlak rough set theory, information system, and so forth. Section 3 mainly explores some extension Pawlak rough set models. Firstly some properties of generalized fuzzy rough set are discussed. The second, the composed fuzzy rough set is initiated and formally defined; its some properties are also discussed in detailed. The four section illustrate the fuzzy RSAR based the composed fuzzy rough set; and an example is given to show its efficiency and accuracy in classification. In section 5 we make a conclusion on the paper.

2 Preliminaries

Let $U= \{u_1, u_2, \ldots, u_n\}$ stands for the finite and nonempty set of objects. The power $P(U)$ can be viewed as a subset of fuzzy power $\mathcal{F}(U)$, $X\in\mathcal{F}(U)$ can be represented as form $X=\{(u, \mu_{X(u)})| u \in U\}$, where for every $u\in U$, the value $\mu_{X(u)}\in[0, 1]$. X is also represented as form $X=(\mu_{X(u1)}, \mu_{X(u2)}, \ldots, \mu_{X(un)})$ when U is finite set, or as $X=\int \mu_{X(u)}/u$ when U is infinite set. For arbitrary $\lambda\in I=[0, 1]$, the λ-level X_λ and the strong λ-level $X_{\lambda+}$ are respectively $X_\lambda=\{u\in U|\mu_{X(u)}\geq\lambda\}$ and $X_{\lambda+}=\{u\in U|\mu_{X(u)}>\lambda\}$, $X= \bigvee_{\lambda\in I} (\lambda \wedge X_\lambda)$ $= \bigvee_{\lambda\in I} (\lambda\wedge X_{\lambda+})$, , $X_0=U$, $X_{1+}=\emptyset$ [7]. A fuzzy binary relation R over U is a function $R: U\times U\rightarrow[0, 1]$, its membership function is represented by $\mu_R(x, y)$. The class of all fuzzy binary relations of U will be denoted as $\mathcal{F}(U\times U)$.

Let R be an ordinary equivalence relation on U, U/R denotes the equivalence classes by R. For every $u\in U$, $[u]_R\in U/R$ denotes the equivalence class of u, the pair (U, R) is called as the Pawlak approximation space. Let $X\subseteq U$,

$$\underline{R}\ X=\{u|[u]_R\subseteq X\},\ \overline{R}\ X=\{u|[u]_R\cap X\neq\emptyset\} \tag{2.1}$$

Where $\underline{R}\ X$ is called the lower approximation of X, while $\overline{R}\ X$ is the upper approximation of X, ($\underline{R}\ X, \overline{R}\ X$) is called as rough set of X. $\alpha_R(X)$ =card($\underline{R}\ X$) /card($\overline{R}\ X$) denotes the accuracy of approximation, $\overline{R}\ X$- $\underline{R}\ X$ as the boundary set of X. The fuzzy set $\widetilde{X}$ over U is defined as following [7]:

$$\mu_{\tilde{X}}(u)=\frac{\mathrm{card}([u]_R \cap X)}{\mathrm{card}([u]_R)}, \text{ for every object } u \in U \tag{2.2}$$

So, $\mu_{\tilde{X}}(u)=1$ iff $u \in \underline{R} X$; $\mu_{\tilde{X}}(u)=0$ iff $u \in U-\overline{R} X$; otherwise $0<\mu_{\tilde{X}}(u)<1$ iff $u \in \overline{R} X-\underline{R} X$. However, if X is fuzzy rough set or R is fuzzy equivalence relation over U, the above calculation formula of membership function $\mu_{\tilde{X}}(u)$ may need some changes in form.

Information system (IS) is an ordered quadruple S=(U, A, f, V), where $U=\{u_1, u_2, \ldots, u_n\}$ is a non-empty finite objects set, $A=\{a_1, a_2, \ldots, a_m\}$ is a non-empty finite attributes set. $V=\cup_{a \in A} V_a$ is a set of attributes values, where V_a is the domain of attribute $a \in A$. $f: U \times A \rightarrow V$ is an information function, where for all $(u, a) \in U \times A$, $f(u, a) \in V_a$, $Inf(u)=\{(a, f_a(u)) | a \in A\}$ is called as an information vector of u.

Let $P \subseteq A$, $Ind(P)=\{(u, v) \in U \times U |$ for all $a \in P$, $f(a, u)=f(a, v)\}$. If $(u, v) \in Ind(P)$, then u and v are indiscernible under attributes subset P. For every $u \in U$, its equivalence class is denoted as $[u]_P=\{v | (u, v) \in Ind(P)\}$ and $U/P=\{[u]_P | u \in U\}$. Let $P, Q \subseteq A$, the positive region $POS_P(Q) = \cup_{X \in U/Q} \underline{P} X$ contains all objects of U that can be classified to classes of U/Q by using the knowledge in attributes P. For $P, Q \subseteq A$, we call Q depends on P in a degree k ($0 \leq k \leq 1$), where

$$k=\gamma_P(Q)=\frac{\mathrm{card}(POS_P(Q))}{\mathrm{card}(U)} \tag{2.3}$$

If k=1, then call Q depends totally on P; if $0<k<1$, then call Q depends partially on P with the degree k, denoted by $P \Rightarrow_k Q$; and if k=0 then call Q does not depend on P.

3 Extensions of Pawlak Rough Set Model

Let $U=\{u_1, u_2, \ldots, u_n\}$ and $W=\{w_1, w_2, \ldots, w_m\}$ be two finite and nonempty sets, $R \in \mathcal{F}(U \times W)$ is fuzzy relation from U to W. When U and W are finite nonempty sets, R can be represented by n*m matrix $R=(r_{ij})_{n \times m}$ where $r_{ij}=\mu_R(u_i, w_j) \in [0, 1]$, for all i=1, 2, …, n, j=1, 2, …, m [3]. For each $\lambda \in [0, 1]$, matrix $(\lambda r_{ij})_{n \times m}$ denotes the cut relation R_λ, where $\lambda r_{ij}=1$ iff $r_{ij} \geq \lambda$, otherwise $\lambda r_{ij}=0$. If $R \in P(U \times W)$, then for all $u \in U$, we call $R_W(u)=\{w \in W | (u, w) \in R\}$ as the successor neighborhood of u.

Definition 1. Let U be two finite and nonempty sets, $R \in \mathcal{F}(U \times W)$. $Y \in \mathcal{F}(W)$, $u \in U$, the generalized fuzzy rough set ($\underline{apr}_R Y$, $\overline{apr}_R Y$) can be defined as following:

$$\underline{\mathrm{apr}}_R\ Y(u)=\min_{w\in W}\{\max(1-R(u,w),Y(w))\}$$
$$\overline{\mathrm{apr}}_R\ Y(u)=\max_{w\in W}\{\min(R(u,w),Y(w))\} \tag{3.1}$$

Where the triple (U, W, R) is called as generalized fuzzy approximation space.

Theorem 1. Let $R\in\mathcal{F}(U\times W)$, then for all $Y\in\mathcal{F}(W)$ and arbitrary $\alpha\in[0,1]$,

$$(\underline{\mathrm{apr}}_R\ Y)_\alpha=\underline{\mathrm{apr}}_{R_{(1-\alpha)+}}\ Y_\alpha,\ (\overline{\mathrm{apr}}_R\ Y)_\alpha=\overline{\mathrm{apr}}_{R_\alpha}\ Y_\alpha$$
$$(\underline{\mathrm{apr}}_R\ X)_{\alpha+}=\underline{\mathrm{apr}}_{R_{(1-\alpha)}}\ Y_{\alpha+},\ (\overline{\mathrm{apr}}_R\ Y)_{\alpha+}=\overline{\mathrm{apr}}_{R_{\alpha+}}\ Y_{\alpha+}$$

Proof. For arbitrary $\alpha\in[0,1]$,

$$\begin{aligned}(\underline{\mathrm{apr}}_R\ Y)_\alpha&=\{u\in U\mid \underline{\mathrm{apr}}_R\ Y(u)\ge\alpha\}\\&=\{u\in U\mid \min_{w\in W}\{\max(1-R(u,w),Y(w))\}\ge\alpha\}\\&=\{u\in U\mid \text{for all } w\in W,\ \max(1-R(u,w),Y(w))\ge\alpha\}\\&=\{u\in U\mid \text{for each } w\in W,\ 1-R(u,w)\ge\alpha,\text{ or } Y(w)\ge\alpha\}\\&=\{u\in U\mid \{w\in W\mid 1-R(u,w)\ge\alpha\}\cup\{w\in W\mid Y(w)\ge\alpha\}=W\}\\&=\{u\in U\mid \{w\in W\mid R(u,w)>1-\alpha\}\subseteq\{w\in W\mid Y(w)\ge\alpha\}\}\\&=\{u\in U\mid (R_W(u))_{(1-\alpha)+}\subseteq Y_\alpha\}=\underline{\mathrm{apr}}_{R_{(1-\alpha)+}}\ Y_\alpha\end{aligned}$$

$$\begin{aligned}(\overline{\mathrm{apr}}_R\ Y)_\alpha&=\{u\in U\mid \overline{\mathrm{apr}}_R\ Y(u)\ge\alpha\}\\&=\{u\in U\mid \max_{w\in W}\{\min(R(u,w),Y(w))\}\ge\alpha\}\\&=\{u\in U\mid \exists\, w\in W,\ \min(R(u,w),Y(w))\ge\alpha\}\\&=\{u\in U\mid \exists\, w\in W,\ R(u,w)\ge\alpha \text{ and } Y(w)\ge\alpha\}\\&=\{u\in U\mid (R_W(u))_\alpha\cap Y_\alpha\neq\emptyset\}=\overline{\mathrm{apr}}_{R_\alpha}\ Y_\alpha\end{aligned}$$

$$\begin{aligned}(\underline{\mathrm{apr}}_R\ Y)_{\alpha+}&=\{u\in U\mid \underline{\mathrm{apr}}_R\ Y(u)>\alpha\}\\&=\{u\in U\mid \min_{w\in W}\{\max(1-R(u,w),Y(w))\}>\alpha\}\\&=\{u\in U\mid \text{for each } w\in W,\ \max(1-R(u,w),Y(w))>\alpha\}\\&=\{u\in U\mid \text{for each } w\in W,\ 1-R(u,w)>\alpha,\text{ or } Y(w)>\alpha\}\\&=\{u\in U\mid \{w\in W\mid 1-R(u,w)>\alpha\}\cup\{w\in W\mid Y(w)>\alpha\}=W\}\\&=\{u\in U\mid \{w\in W\mid R(u,w)\ge 1-\alpha\}\subseteq\{w\in W\mid Y(w)>\alpha\}=W\}\\&=\{u\in U\mid (R_W(u))_{1-\alpha}\subseteq Y_{\alpha+}\}=\underline{\mathrm{apr}}_{R_{(1-\alpha)}}\ Y_{\alpha+}\end{aligned}$$

$$\begin{aligned}(\overline{\mathrm{apr}}_R\ Y)_{\alpha+}&=\{u\in U\mid \overline{\mathrm{apr}}_R\ Y(u)>\alpha\}\\&=\{u\in U\mid \max_{w\in W}\{\min(R(u,w),Y(w))\}>\alpha\}\end{aligned}$$

$$= \{ u \in U \mid \exists w \in W, \min(R(u, w), Y(w))>\alpha\}$$
$$= \{ u \in U \mid \exists w \in W, R(u, w)>\alpha \text{ and } Y(w)>\alpha\}$$
$$= \{ u \in U \mid (R_W(u))_{\alpha+} \cap Y_{\alpha+} \neq \emptyset\} = \overline{apr}_{R_{\alpha+}} Y_{\alpha+}$$

Remark 1. From the theorem 1 and the decompose theory, we can immediately conclude that the following conclusion.

1) $\underline{apr}_R Y = \bigvee_{\alpha\in[0,1]} (\alpha \bigwedge (\underline{apr}_{R_{(1-\alpha)+}} Y_\alpha)) = \bigvee_{\alpha\in[0,1]} (\alpha \bigwedge (\underline{apr}_{R_{(1-\alpha)}} Y_{\alpha+}))$

2) $\overline{apr}_R Y = \bigvee_{\alpha\in[0,1]} (\alpha \bigwedge (\overline{apr}_{R_{\alpha+}} Y_{\alpha+})) = \bigvee_{\alpha\in[0,1]} (\alpha \bigwedge (\overline{apr}_{R_\alpha} Y_\alpha))$

In paper [6], Wu construct the generalized fuzzy rough sets exactly starting from the above formulas 1) and 2). Our results in the paper show these two approaches are totally equivalence.

Let the fuzzy equivalence classes set $U/R=\{F_1, F_2, \ldots, F_k\}$, we consider the approximations problem for every $X \in \mathcal{F}(U)$, let $\overline{Apr}_R X$ denotes upper approxim -ation and $\underline{Apr}_R X$ lower approximation respectively. Because the membership values of individual object to the approximations are not explicitly available directly, so we need obtain them from another point. Let $F_i \in U/R$ denoted by $F_i=\{(u, \mu_{Fi(u)}) | u \in U\}$, consider the following fuzzy set forms:

$$\underline{Apr}_R X = \sum_{i=1,\cdots,k} \mu_{\underline{RX}}(F_i)/F_i \quad \overline{Apr}_R X = \sum_{i=1,\cdots,k} \mu_{\overline{RX}}(F_i)/F_i \tag{3.2}$$

Where $\underline{RX}$ and $\overline{RX}$ are short writing of fuzzy sets $\underline{Apr}_R X$ and $\overline{Apr}_R X$, respectively. For every $F \in U/R= \{F_1, F_2, \ldots, F_k\}$, the membership degree values $\mu_{\underline{RX}}(F)$ and $\mu_{\overline{RX}}(F)$ can be respectively calculated by the following:

$$\mu_{\underline{RX}}(F)= \min_{v\in U} \max (1-\mu_F(v), X(v)), \quad \mu_{\overline{RX}}(F)= \max_{v\in U} \min (\mu_F(v), X(v)).$$

On the other hand, for every $u \in U$,

$$\mu_{\underline{RX}}(u)= \max_{F\in U/R} \min (\mu_F(u), \mu_{\underline{RX}}(F)), \quad \mu_{\overline{RX}}(u)= \max_{F\in U/R} \min (\mu_F(u), \mu_{\overline{RX}}(F)).$$

If R is a crisp equivalence relation over U and $U/R=\{F_1, F_2, \ldots, F_k\}$. Let $X \in P(U)$, then the $\mu_{\underline{RX}}(F)$ and $\mu_{\overline{RX}}(F)$ can be computed as follows.

$$\mu_{\underline{RX}}(F)= \min_{v\in U} \max (1-\mu_F(v), X(v)) = \min_{v\in F} X(v)$$
$$\mu_{\overline{RX}}(F)= \max_{v\in U} \min (\mu_F(v), X(v)) = \max_{v\in F} X(v)$$

It means that $\mu_{\underline{RX}}(F)=1$ iff $F\subseteq X$, otherwise $\mu_{\underline{RX}}(F)=0$; and $\mu_{\overline{RX}}(F)=1$ iff $F\cap X\neq\emptyset$, other wise $\mu_{\overline{RX}}(F)=0$. Therefore,

$$\mu_{\underline{RX}}(u)=\max_{F\in U/R}\min(\mu_F(u),\ \mu_{\underline{RX}}(F))=\max\{\mu_F(u)\mid u\in F \text{ and } F\subseteq X\}$$
$$\mu_{\overline{RX}}(u)=\max_{F\in U/R}\min(\mu_F(u),\ \mu_{\overline{RX}}(F))=\max\{\mu_F(u)\mid u\in F \text{ and } F\cap X\neq\emptyset\}$$

Above statements show that our viewpoint is a natural generalization of Pawlak rough set from crisp case to the fuzzy circumstance. Below the paper, we will present the more general statement of above extension. Let $U=\{u_1, u_2, \ldots, u_n\}$ and $W=\{w_1, w_2, \ldots, w_m\}$ be two finite and nonempty universe,

$$R=\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{pmatrix}=(r_{ij})_{n\times m}=\begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix}=(r'_1, r'_2,\ldots, r'_m) \tag{3.3}$$

Where for all i=1, 2, …, n, j=1, 2, …, m, $r_{ij}=\mu_R(u_i, w_j)$, $R_W(u_i)=r_i=(r_{i1}, r_{i2}, \ldots, r_{im})$, $R_U(w_j)=r'_j=(r_{1j}, r_{2j}, \ldots, r_{nj})^T$, $R'=(r'_{ij})_{m\times n}\in\mathcal{F}(W\times U)$, where $r'_{ji}=r_{ij}$.

Definition 2. Let U and W be two finite and nonempty sets, $R\in\mathcal{F}(U\times W)$. $X\in\mathcal{F}(W)$, then $\overline{\mathrm{Apr}}_R X$, $\underline{\mathrm{Apr}}_R X\in\mathcal{F}(W)$ can be defined as following. For all $v\in W$,

$$\underline{\mathrm{Apr}}_R X(v)=\max_{u\in U}\min(R'(v, u),\ \min_{w\in W}\max(1-R(u, w), X(w))),$$
$$\overline{\mathrm{Apr}}_R X(v)=\max_{u\in U}\min(R'(v, u),\ \max_{w\in W}\min(R(u, w), X(w))) \tag{3.4}$$

The pair ($\underline{\mathrm{Apr}}_R X$, $\overline{\mathrm{Apr}}_R X$) is called as the composed fuzzy rough set of X, and the triple (U, W, R) as the composed fuzzy approximation space.

Proposition 4. Let U and W be two finite and nonempty universes, $R\in\mathcal{F}(U\times W)$. Then for all $X\in\mathcal{F}(W)$,

$$\underline{\mathrm{Apr}}_R X=\overline{\mathrm{apr}}_{R'}(\underline{\mathrm{apr}}_R X),\ \overline{\mathrm{Apr}}_R X=\overline{\mathrm{apr}}_{R'}(\overline{\mathrm{apr}}_R X).$$

Where $\underline{\mathrm{apr}}_R$ and $\overline{\mathrm{apr}}_R$ are respectively generalized fuzzy rough lower and upper approximation operators, $\overline{\mathrm{apr}}_{R'}$ is upper approximation operator related with R'.

If U=W and R is a fuzzy equivalence relation, then R=R' and R' (v, u) =R (u, v). For each $u \in U$, $F_u = \sum_{v \in U} R(u, v)/v$. Furthermore,

$$\underline{Apr}_R X = \sum_{u \in U} (\min_{v \in U} \max(1-R(u, v), X(v)))/F_u$$
$$\overline{Apr}_R X = \sum_{u \in U} (\max_{v \in U} \min(R(u, v), X(v)))/F_u \quad (3.5)$$

Therefore, from above definition 2,

$$\underline{Apr}_R X(v) = \max_{u \in U} \min(R'(v, u), \min_{w \in W} \max(1-R(u, w), X(w)))$$
$$= \max_{u \in U} \min(\mu_{Fu}(v), \mu_{\underline{RX}}(F_u))$$
$$\overline{Apr}_R X(v) = \max_{u \in U} \min(R'(v, u), \max_{w \in W} \min(R(u, w), X(w)))$$
$$= \max_{u \in U} \min(\mu_{Fu}(v), \mu_{\overline{RX}}(F))$$

Proposition 5. Let U be finite and nonempty set, $R \in P(U \times U)$, then for all $X \in P(U)$,

$$\underline{Apr}_R X = \underline{R}\, X, \quad \overline{Apr}_R X = \overline{R}\, X$$

Proof. Since $R \in P(U \times U)$, $X \in P(U)$, then for arbitrary $u \in U$, there exists a unique equivalence class $[u]_R = \sum_{v \in U} R(u, v)/v$. Therefore, for all $X \in P(U)$ and $u \in U$,

$$\underline{Apr}_R X(v) = \max_{u \in U} \min(R'(v, u), \min_{w \in W} \max(1-R(u, w), X(w)))$$
$$= \max_{u \in U} \min(R(v, u), \min(\min_{w \notin [u]_R} \max(1-R(u, w), X(w)), \min_{w \in [u]_R} \max(1-R(u, w), X(w))))$$
$$= \max_{u \in U} \min(R(v, u), \min_{w \in [u]_R} X(w))$$
$$= \max(\max_{u \in [v]_R} \min(R(v, u), \min_{w \in [u]_R} X(w)), \max_{u \notin [v]_R} \min(R(v, u), \min_{w \in [u]_R} X(w)))$$
$$= \min_{w \in [v]_R} X(w) = \underline{R}\, X(v)$$

$$\overline{Apr}_R X(v) = \max_{u \in U} \min(R'(v, u), \max_{w \in W} \min(R(u, w), X(w)))$$
$$= \max_{u \in U} \min(R(v, u), \max(\max_{w \in [u]_R} \min(R(u, w), X(w)), \max_{w \notin [u]_R} \min(R(u, w), X(w))))$$
$$= \max_{u \in U} \min(R(v, u), \max_{w \in [u]_R} X(w))$$
$$= \max(\max_{u \in [v]_R} \min(R(v, u), \max_{w \in [u]_R} X(w)), \max_{u \notin [v]_R} \min(R(v, u), \max_{w \in [u]_R} X(w)))$$
$$= \max_{w \in [v]_R} X(w) = \overline{R}\, X(v).$$

That is $\underline{\mathrm{Apr}}_R X=\underline{R}\ X$, $\overline{\mathrm{Apr}}_R X=\overline{R}\ X$.

Theorem 2. Let $R\in\mathcal{F}(U\times W)$, then for all $X\in\mathcal{F}(W)$, and arbitrary $\alpha\in[0, 1]$,

1) $(\underline{\mathrm{Apr}}_R X)_\alpha=\overline{\mathrm{apr}}_{(R')_\alpha}\ \underline{\mathrm{apr}}_{R_{(1-\alpha)+}} X_\alpha$, 2) $(\overline{\mathrm{Apr}}_R X)_\alpha=\overline{\mathrm{apr}}_{(R')_\alpha}\ \overline{\mathrm{apr}}_{R_\alpha} X_\alpha$,

3) $(\underline{\mathrm{Apr}}_R X)_{\alpha+}=\overline{\mathrm{apr}}_{(R')_{\alpha+}}\ \underline{\mathrm{apr}}_{R_{(1-\alpha)}} X_{\alpha+}$, 4) $(\overline{\mathrm{Apr}}_R X)_{\alpha+}=\overline{\mathrm{apr}}_{(R')_{\alpha+}}\ \overline{\mathrm{apr}}_{R_{\alpha+}} X_{\alpha+}$.

Proof. For arbitrary $\alpha\in[0, 1]$,

1) $(\underline{\mathrm{Apr}}_R X)_\alpha=\{v\in U\mid \underline{\mathrm{Apr}}_R X(v)\geq\alpha\}$

$=\{v\in U\mid \max_{u\in U}\min(R'(v, u),\ \min_{w\in W}\max(1-R(u, w), X(w)))\geq\alpha\}$

$=\{v\in U\mid \exists u\in U, R'(v, u)\geq\alpha \text{ and } \min_{w\in W}\max(1-R(u, w), X(w))\geq\alpha\}$

$=\{v\in U\mid (R'_v)_\alpha\cap(\underline{\mathrm{apr}}_R X)_\alpha\neq\emptyset\}=\overline{\mathrm{apr}}_{(R')_\alpha}(\underline{\mathrm{apr}}_R X)_\alpha=\overline{\mathrm{apr}}_{(R')_\alpha}\ \underline{\mathrm{apr}}_{R_{(1-\alpha)+}} X_\alpha$

2) $(\overline{\mathrm{Apr}}_R X)_\alpha=\{v\in U\mid \overline{\mathrm{Apr}}_R X(v)\geq\alpha\}$

$=\{v\in U\mid \max_{u\in U}\min(R'(v, u),\ \max_{w\in W}\min(R(u, w), X(w)))\geq\alpha\}$

$=\{v\in U\mid \exists u\in U, R'(v, u)\geq\alpha \text{ and } \max_{w\in W}\min(R(u, w), X(w))\geq\alpha\}$

$=\{v\in U\mid (R'_v)_\alpha\cap(\overline{\mathrm{apr}}_R X)_\alpha\neq\emptyset\}=\overline{\mathrm{apr}}_{(R')_\alpha}(\overline{\mathrm{apr}}_R X)_\alpha=\overline{\mathrm{apr}}_{(R')_\alpha}\ \overline{\mathrm{apr}}_{R_\alpha} X_\alpha$

3) $(\underline{\mathrm{Apr}}_R X)_{\alpha+}=\{v\in U\mid \underline{\mathrm{Apr}}_R X(u)>\alpha\}$

$=\{v\in U\mid \max_{u\in U}\min(R'(v, u),\ \min_{w\in W}\max(1-R(u, w), X(w)))>\alpha\}$

$=\{v\in U\mid \exists u\in U, R'(v, u)>\alpha \text{ and } \min_{w\in W}\max(1-R(u, w), X(w))>\alpha\}$

$=\{v\in U\mid (R'_v)_{\alpha+}\cap(\underline{\mathrm{apr}}_R X)_{\alpha+}\neq\emptyset\}=\overline{\mathrm{apr}}_{(R')_{\alpha+}}(\underline{\mathrm{apr}}_R X)_{\alpha+}=\overline{\mathrm{apr}}_{(R')_\alpha}\ \underline{\mathrm{apr}}_{R_{(1-\alpha)}} X_{\alpha+}$

4) $(\overline{\mathrm{Apr}}_R X)_{\alpha+}=\{v\in U\mid \overline{\mathrm{Apr}}_R X(v)>\alpha\}$

$=\{v\in U\mid \max_{u\in U}\min(R'(v, u),\ \max_{w\in W}\min(R(u, w), X(w)))>\alpha\}$

$=\{v\in U\mid \exists u\in U, (R'(v, u)>\alpha \text{ and } \max_{w\in W}\min(R(u, w), X(w))>\alpha)$

$=\{v\in U\mid (R'_v)_{\alpha+}\cap(\overline{\mathrm{apr}}_R X)_{\alpha+}\neq\emptyset\}=\overline{\mathrm{apr}}_{(R')_{\alpha+}}(\overline{\mathrm{apr}}_R X)_{\alpha+}=\overline{\mathrm{apr}}_{(R')_{\alpha+}}\ \overline{\mathrm{apr}}_{R_{\alpha+}} X_{\alpha+}$

Remark 2. From above theorem 2, then

1) $\underline{\mathrm{Apr}}_R Y=\bigvee_{\alpha\in[0,1]}(\alpha\bigwedge(\overline{\mathrm{apr}}_{(R')_\alpha}\ \underline{\mathrm{apr}}_{R_{(1-\alpha)+}} Y_\alpha))=\bigvee_{\alpha\in[0,1]}(\alpha\bigwedge(\overline{\mathrm{apr}}_{(R')_\alpha}\ \underline{\mathrm{apr}}_{R_{(1-\alpha)}} X_{\alpha+}))$,

2) $\overline{\mathrm{Apr}}_R Y=\bigvee_{\alpha\in[0,1]}(\alpha\bigwedge(\overline{\mathrm{apr}}_{(R')_\alpha}\ \overline{\mathrm{apr}}_{R_\alpha} X_\alpha))=\bigvee_{\alpha\in[0,1]}(\alpha\bigwedge(\overline{\mathrm{apr}}_{(R')_{\alpha+}}\ \overline{\mathrm{apr}}_{R_{\alpha+}} X_{\alpha+}))$.

4 Attribute Reduction Based on Composed Fuzzy-Rough Set

An information system S=(U, A, f, V) is called as decision table system, if $A=C \cup D$ and $C \cap D=\varnothing$, where C is the conditional attributes set and $D \neq \varnothing$ is the set of decision attributes. The issue of decision table mainly focuses on how to obtain whole rules by as less rules and attributes as possible from the information table. The main approach is attribute reduction including the reduction of attributes-values and deletion of redundant rules. In fuzzy case, fuzzy rough set attributes reduction (Fuzzy RSAR) should be built on the notion of the composed fuzzy lower approximation. Let fuzzy decision table system S=(U, $C \cup \{d\}$, f, V), for arbitrary $P \subseteq C$, the fuzzy positive region $POS_P(\{d\})= \bigcup_{F \in U/\{d\}} \underline{Apr}_P F$, where $\mu_{POS_P(\{d\})}(u) = \sup_{F \in U/\{d\}} \mu_{\underline{PF}}(u)$, Then the dependency function $\gamma_P(\{d\})$ can be calculated by the following:

$$\gamma_P(\{d\})=\frac{|POS_P(\{d\})|}{|U|}=\frac{\Sigma_{u \in U} \mu_{POS_P(\{d\})}(u)}{|U|} \tag{4.1}$$

In fuzzy case, we use the fuzzy positive region $POS_C(\{d\})$ rather than |U| as the denominator of normalization, then

$$\gamma(\{d\})=\frac{|POS_P(\{d\})|}{|POS_C(\{d\})|}=\frac{\Sigma_{u \in U} \mu_{POS_P(\{d\})}(u)}{\Sigma_{u \in U} \mu_{POS_C(\{d\})}(u)} \tag{4.2}$$

An data set example from stock market [10] is given to illustrate the operation of fuzzy RSAR, in which $U=\{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$, two real-valued attributes are feature a (profit ratio of per stock) and feature b (harvest ratio of per capital), decision 2-valued attribute is d (representing invest or not), fuzzy equivalence class over U are (H_a, L_a) partitioned by attribute a and (H_b, L_b) partitioned by attribute b, respectively.

Table 1. Stock Information Table

U	a: profit ratio		b: harvest ratio of per capital		d: investment
	L_a	H_a	L_b	H_b	
1	1	0	1	0	Y
2	0.7	0.3	0.2	0.8	N
3	0.8	0.2	0.9	0.1	Y
4	0.9	0.1	1	0	Y
5	0.1	0.9	0.2	0.8	N
6	0.8	0.2	1	0	Y
7	0.1	0.9	0.2	0.8	N
8	0.8	0.2	0.2	0.8	Y

Setting A={a}, B={b}, C={a, b} and Q={d}, then the following equivalence classes are obtained from the above decision table.

$$U/Q=\{X, Y\}=\{\{u_1, u_3, u_4, u_6, u_8\}, \{u_2, u_5, u_7\}\}.$$

$U/A= \{L_a=(1.0,0.7,0.8,0.9,0.1,0.8,0.1,0.8), H_a= (0.0,0.3,0.2,0.1,0.9,0.2,0.9,0.2)\}$,

$U/B= \{L_b= (1.0,0.2,0.9,1.0,0.2,1.0,0.2,0.2), H_b= (0.0,0.8,0.1,0.0,0.8,0.0,0.8,0.8)\}$,

$U/C= \{L_a \cap L_b, L_a \cap H_b, H_a \cap L_b, H_a \cap H_b\}$

$= \{(1.0, 0.2, 0.8, 0.9, 0.1, 0.8, 0.1, 0.2), (0.0, 0.7, 0.1, 0.0, 0.1, 0.0, 0.1, 0.8),$

$(0.0, 0.2, 0.2, 0.1, 0.2, 0.2, 0.2, 0.2), (0.0, 0.3, 0.1, 0.0, 0.8, 0.0, 0.8, 0.2)\}$.

The first step is to calculate the lower approximations of the sets A, B and C. For simplicity, only A will be considered here. For object u_1 and decision equivalence class $X= \{u_1, u_3, u_4, u_6, u_8\}$ and $Y=\{u_2, u_5, u_7\}$,

$$\mu_{\underline{AX}}(u_1)= \max_{F \in U/A} \min (\mu_F(u_1), \min_{y \in U} \max \{1-\mu_F(y), \mu_X(y)\})=0.3$$

$$\mu_{\underline{AY}}(u_1)= \max_{F \in U/A} \min (\mu_F(u_1), \min_{y \in U} \max \{1-\mu_F(y), \mu_X(y)\})=0.0$$

Hence, $\mu_{POS_A(Q)}(u_1) =0.3$. For the other objects, $\mu_{POS_A(Q)}(u_2)=0.3$, $\mu_{POS_A(Q)}(u_3) =0.3$, $\mu_{POS_A(Q)}(u_4) =0.3$, $\mu_{POS_A(Q)}(u_5)=0.8$, $\mu_{POS_A(Q)}(u_6)=0.2$, $\mu_{POS_A(Q)}(u_7)=0.8$, $\mu_{POS_A(Q)}(u_8)=0.3$. Then $r_A(Q)= \frac{\Sigma_{u \in U} \mu_{POS_A(Q)}(u)}{|U|} =3.3/8=0.4125$. Calculating for B and C gives $r_B(Q)=5/8=0.625$, $r_C(Q)=5.4/8=0.675$. Because there are only two condition attributes in this example, so its core and reduction are set {a, b}. The result is exactly in accordance with that of come from by the method of fuzzy cluster in paper [10].

5 Conclusions

As a suitable mathematical model to handle partial knowledge in data set, traditional RSAR encounters some critical problems when the noise and real-valued attributes value is included in the information system. The fuzzy RSAR method can alleviate these important problems and has been applied in more than one field with very promising results. In this paper, we study the fuzzy RSAR methods used in fuzzy information systems. We extend the rough set model to fuzzy case and present the formal definition of the composed fuzzy rough set. We also illustrate the fuzzy RSAR method and give a simple example to show its higher efficiency and accuracy.

References

1. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. International Journal General Systems 17(2-3), 191–209 (1990)
2. Jensen, R., Shen, Q.: Fuzzy rough attribute reduction with application to web categorization. Fuzzy Sets and Systems 141, 469–485 (2004)

3. Liu, G.L.: The Axiomatic Systems of Rough Fuzzy Sets on Fuzzy Approximation Space (in Chinese). Chinese Journal Of Computers 27(9), 1187–1191 (2004)
4. Pawlak, Z.: Rough sets. In: Lin, T.Y., Cercone, N. (eds.) Rough Sets and Data Mining, pp. 3–8. Kluwer Academic Publishers, Boston (1997)
5. Slowinski, R.: Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, pp. 287–304. Kluwer Academic Publishers, Dordrecht (1992)
6. Wu, W.-Z., Mi, J.-S., Zhang, W.-X.: Generalized fuzzy rough sets. Information Sciences 151, 263–282 (2003)
7. Zhang, W., Wu, W., Liang, J., Li, D.: Theory and Methods of the Rough Sets (in Chinese). Sciences Publisher Press, Beijing (2001)
8. Hong, X., Zhong, C.H., Xiao, N.D.: Discretization of Continuous Attributes in Rough Set Theory Based on Information Entropy(in Chinese). Chinese Journal of Computers 28(9), 1570–1573 (2005)
9. Yao, Y.Y.: A comparative study of fuzzy sets and rough sets. J. of Information Science 109(1-4), 227–242 (1998)
10. Zhang, S., Sun, J., Zhang, J.: A Study on the Reducing Method of Rough Set Decision Table Based on Fuzzy Cluster (in Chinese). Computer Engineering and Applications 15, 175–177 (2004)

Author Index