

Jukka Riecki  
Mika Ylianttila  
Minyi Guo (Eds.)

LNCS 6646

# Advances in Grid and Pervasive Computing

6th International Conference, GPC 2011  
Oulu, Finland, May 2011  
Proceedings



Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*TU Dortmund University, Germany*

Madhu Sudan

*Microsoft Research, Cambridge, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbruecken, Germany*

Jukka Riekkı Mıka Ylıanttıla Mınıyı Guo (Eds.)

# Advances in Grid and Pervasive Computing

6th International Conference, GPC 2011  
Oulu, Finland, May 11-13, 2011  
Proceedings

## Volume Editors

Jukka Riekk  
University of Oulu  
Department of Electrical and Information Engineering  
90014 Oulu, Finland  
E-mail: jukka.riekki@oulu.fi

Mika Ylianttila  
University of Oulu  
Department of Electrical and Information Engineering  
90014 Oulu, Finland  
E-mail: mika.ylianttila@oulu.fi

Minyi Guo  
Shanghai Jiao Tong University  
Department of Computer Science and Engineering  
Minhang, Shanghai, 200240, China  
E-mail: guo-my@cs.sjtu.edu.cn

ISSN 0302-9743  
ISBN 978-3-642-20753-2  
DOI 10.1007/978-3-642-20754-9  
Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349  
e-ISBN 978-3-642-20754-9

Library of Congress Control Number: 2011925941

CR Subject Classification (1998): F.2, C.2, H.4, D.2, D.4, C.2.4

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

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

Grid and Pervasive Computing (GPC) is an annual international conference on the emerging areas of grid computing, cloud computing, and pervasive computing. The 6th International Conference on Grid and Pervasive Computing, GPC 2011, was held in Oulu, Finland, during May 11-13, 2011. This volume contains the full papers that were presented at the conference. This program was preceded by one day of workshops, a doctoral colloquium and tutorials. The workshop and doctoral colloquium papers were published in a separate volume after the conference.

We received 62 submissions originating from 19 countries. Each submission was reviewed by at least 3, and on average 3.3, Program Committee members. Finally, the Program Committee selected 28 full papers for presentation at the conference and inclusion in this LNCS volume. The selected papers present a cross-section of research being carried out and the recent trends in the fields of grid, cloud, and pervasive computing. The wide range of topics illustrates the variety of challenges that need to be tackled in these fields of research.

The progress in these fields is fast, as shown by the keynote on how applying computer science tools and technologies - like cloud computing - to breakthrough science is accelerating scientific progress. The first tutorial provided a view to the UBI program that is building a functional prototype of an open ubiquitous city, by deploying new pervasive computing infrastructure such as public displays and wireless networks at downtown Oulu, and employing the infrastructure to provide novel prototype services to the citizens. The other two tutorials showed how volunteer computing platforms can be built with the XtremWeb-CH middleware and how interoperable multimedia services can be developed using the CAM4Home open platform.

Two workshops were held in conjunction with the GPC 2011 conference: The International Workshop on Health and Well-Being Technologies and Services for Elderly (HWTS 2011) and the International Workshop on Self-Managing Solutions for Smart Environments (S3E 2011). In addition, PhD students received valuable feedback for their doctoral studies in the doctoral colloquium from the colloquium panelists and their peers.

The conference would not have been possible without the support of many people and organizations that helped in various ways to make it a success. The EasyChair conference management system facilitated the review and building this volume. In particular, we would like to thank Infotech Oulu and the MOTIVE program of the Academy of Finland for their financial support. We

are also grateful to the Program Committee members and the external reviewers for their dedication in reviewing the submissions. We also thank the authors for their efforts in writing and revising their papers, and we thank Springer for publishing the proceedings.

May 2011

Jukka Rieki  
Mika Ylianttila  
Minyi Guo

# Organization

## Steering Committee

Hai Jin (Chair)	Huazhong University of Science and Technology, China
Nabil Abdennadher	University of Applied Sciences, Switzerland
Christophe Cerin	University of Paris XIII, France
Sajal K. Das	The University of Texas at Arlington, USA
Jean-Luc Gaudiot	University of California - Irvine, USA
Kuan-Ching Li	Providence University, Taiwan
Cho-Li Wang	The University of Hong Kong, China
Chao-Tung Yang	Tunghai University, Taiwan

## Conference Chairs

Jukka Riekki	University of Oulu, Finland
Depei Qian	Beihang University, China
Mika Ylianttila	University of Oulu, Finland

## Program Chairs

Jukka Riekki	University of Oulu, Finland
Timo Korhonen	Aalto University, Finland
Minyi Guo	Shanghai Jiaotong University, China

## Workshop and Tutorial Chairs

Jiehan Zhou	University of Oulu, Finland
Mika Rautiainen	University of Oulu, Finland
Zhonghong Ou	Aalto University, Finland

## Publication Chairs

Junzhao Sun	University of Oulu, Finland
Zhiwen Yu	Northwestern Polytechnical University, China

## Local Arrangements Chairs

Mika Rautiainen	University of Oulu, Finland
Susanna Pirttikangas	University of Oulu, Finland
Janne Haverinen	University of Oulu, Finland

**Program Committee**

Luciana Arantes	LIP6, France
Michael Beigl	Karlsruhe Institute of Technology, Germany
Ioan Marius Bilasco	University of Science and Technology of Lille, France
Hsi-Ya Chang	National Center for High-Performance Computing, Taiwan
Jiann-Liang Chen	National Taiwan University of Science and Technology, Taiwan
Yuanfang Chen	Dalian University of Technology, China
Cheng-Chin Chiang	National Dong Hwa University, Taiwan
Hao-Hua Chu	National Taiwan University, Taiwan
Yeh-Ching Chung	National Tsing Hua University, Taiwan
Der-Jiunn Deng	National Changhua University of Education, Taiwan
Belli Fevzi	University of Paderborn, Germany
Patrik Floreen	University of Helsinki, Finland
Kaori Fujinami	Tokyo University of Agriculture and Technology, Japan
Dan Grigoras	University College Cork, Ireland
Bin Guo	Institute TELECOM SudParis, France
Janne Haverinen	University of Oulu, Finland
Michael Hobbs	Deakin University, Australia
Hung-Chang Hsiao	National Cheng Kung University, Taiwan
Sun-Yuan Hsieh	National Cheng Kung University, Taiwan
Ching-Hsien (Robert) Hsu	Chung Hua University, Taiwan
Kuo-Chan Huang	National Taichung University, Taiwan
Eero Hyvönen	Aalto University, Finland
Abawajy Jemal	Deakin University, Australia
Qun Jin	Waseda University, Japan
Fahim Kawsar	Bell Labs and University of Lancaster, UK
Shin'ichi Konomi	Tokyo Denki University, Japan
Gerd Kortuem	Lancaster University, UK
Pangfeng Liu	National Taiwan University, Taiwan
Shou-Chih Lo	National Dong Hwa University, Taiwan
Pierre Manneback	University of Mons, Belgium
Rodrigo F. de Mello	University of Sao Paulo, Brazil
Tommi Mikkonen	Tampere University of Technology, Finland
Martti Mäntylä	Helsinki Institute for Information Technology, Finland
Henning Müller	University of Applied Sciences, Switzerland
Tatsuo Nakajima	Waseda University, Japan
Jin Nakazawa	Keio University, Japan
Petteri Nurmi	University of Helsinki, Finland
Masayoshi Ohashi	ATR, Japan



Junjie Peng	Shanghai University, China
Sheng-Lung Peng	National Dong Hwa University, Taiwan
Dana Petcu	Western University of Timisoara, Romania
Susanna Pirttikangas	University of Oulu, Finland
Mika Rautiainen	University of Oulu, Finland
Hedda Schmidtke	TecO, Karlsruhe Institute of Technology, Germany
Junzhao Sun	University of Oulu, Finland
Kazunori Takashio	Keio University, Japan
Stewart Tansley	Microsoft Research, USA
Sasu Tarkoma	University of Helsinki, Finland
Niwat Thepvilojanapong	Mie University, Japan
Reen-Cheng Wang	National Taitung University, Taiwan
Jan-Jan Wu	Academia Sinica, Taiwan
Shiow-Yang Wu	National Dong Hwa University, Taiwan
Jingling Xue	University of New South Wales, Australia
Yuhong Yan	Concordia University, Canada
Takuro Yonezawa	Keio University, Japan
Chen Yu	Huazhong University of Science and Technology, China
Zhiwen Yu	Northwestern Polytechnical University, China
Guoying Zhao	University of Oulu, Finland
Jiehan Zhou	University of Oulu, Finland
Jingyu Zhou	Shanghai Jiaotong University, China
Yuezhi Zhou	Tsinghua University, China

## External Reviewers

Heikki Ailisto	VTT Oulu, Finland
Ammar Alazab	Deakin University, Australia
Sourav Bhattacharya	Helsinki Institute for Information Technology, Finland
Chun-An Chen	National Cheng Kung University, Taiwan
Chia-Wen Cheng	National Chung-Hsing University, Taiwan
Marta Cortés	University of Oulu, Finland
Oleg Davidyuk	University of Oulu, Finland
Kate Gilman	University of Oulu, Finland
Kimmo Halunen	University of Oulu, Finland
Jari Hannuksela	University of Oulu, Finland
Erkki Harjula	University of Oulu, Finland
Simo Hosio	University of Oulu, Finland
Kuo-Chan Huang	National Taichung University of Education, Taiwan
Marko Jurmu	University of Oulu, Finland
Ilmari Juutilainen	University of Oulu, Finland
Otso Kassinen	University of Oulu, Finland

Hiroaki Kimura	Waseda University, Japan
Timo Koskela	University of Oulu, Finland
Hannu Kukka	University of Oulu, Finland
Teemu Leppanen	University of Oulu, Finland
Mika Oja	University of Oulu, Finland
Nick Patterson	Deakin University, Australia
Mikko Perttunen	University of Oulu, Finland
Mikko Polojärvi	University of Oulu, Finland
Juha Röning	University of Oulu, Finland
Antti Tapani Siirtola	University of Oulu, Finland
Jaakko Suutala	University of Oulu, Finland
Iván Sánchez	University of Oulu, Finland
Satu Tamminen	University of Oulu, Finland
Chia-Chen Wei	National Cheng Kung University, Taiwan
Tetsuo Yamabe	Waseda University, Japan

# Table of Contents

## Keynote

Applying Microsoft Research Technologies to the 4th Paradigm in Scientific Research .....	1
<i>Daron G. Green</i>	

## Cloud, Cluster and Grid Computing

Job Status Prediction – Catch Them Before They Fail .....	3
<i>Igor Grudenic and Nikola Bogunovic</i>	
Group-Based Gossip Multicast Protocol for Efficient and Fault Tolerant Message Dissemination in Clouds .....	13
<i>JongBeom Lim, JongHyuk Lee, SungHo Chin, and HeonChang Yu</i>	
Co-management of Power and Performance in Virtualized Distributed Environments .....	23
<i>Mohsen Sharifi, Mahsa Najafzadeh, and Hadi Salimi</i>	
A Leasing Instances Based Billing Model for Cloud Computing .....	33
<i>Qin Yuan, Zhixiang Liu, Junjie Peng, Xing Wu, Jiandun Li, Fangfang Han, Qing Li, Wu Zhang, Xinjin Fan, and Shengyuan Kong</i>	
A Scalable Multiprocessor Architecture for Pervasive Computing .....	42
<i>Long Zheng, Yanchao Lu, Jingyu Zhou, Minyi Guo, Hai Jin, Song Guo, Yao Shen, Jiehan Zhou, and Jukka Riekkii</i>	

## Peer-to-Peer Computing

Enhancing the Reliability of SIP Service in Large-Scale P2P-SIP Networks .....	52
<i>Fei Xu, Hai Jin, Xiaofei Liao, and Fei Qiu</i>	
A Load Balanced Two-Tier DHT with Improved Lookup Performance of Non-popular Data Items .....	62
<i>Mayank Pandey and Banshi Dhar Chaudhary</i>	
Gridlet Economics: Resource Management Models and Policies for Cycle-Sharing Systems .....	72
<i>Pedro Oliveira, Paulo Ferreira, and Luis Veiga</i>	

## Applications and HCI

Anatomy of Automatic Mobile Carbon Footprint Calculator . . . . .	84
<i>Ville Könönen, Miikka Ermes, Jussi Liikka, Arttu Lämsä, Timo Rantalainen, Harri Paloheimo, and Jani Mäntyjärvi</i>	
Empowering Elderly End-Users for Ambient Programming: The Tangible Way . . . . .	94
<i>Johan Criel, Marjan Geerts, Laurence Claeys, and Fahim Kawsar</i>	
Prototyping Augmented Traditional Games: Concept, Design and Case Studies . . . . .	105
<i>Tetsuo Yamabe, Takahiro Iwata, Takahiro Shichinohe, and Tatsuo Nakajima</i>	

## Modelling and Verification

Modeling and Experimental Validation of the Data Handover API . . . . .	117
<i>Soumeya Leila Hernane, Jens Gustedt, and Mohamed Benyettou</i>	
Formal Modelling and Initial Validation of the Chelonia Distributed Storage System . . . . .	127
<i>Sami Taktak and Lars M. Kristensen</i>	
An Integrated Network Scanning Tool for Attack Graph Construction . . . . .	138
<i>Feng Cheng, Sebastian Roschke, and Christoph Meinel</i>	

## Service Architectures

Context-Awareness Micro-architecture for Smart Spaces . . . . .	148
<i>Susanna Pantsar-Syväniemi, Jarkko Kuusijärvi, and Eila Ovaska</i>	
Distributed Web Service Architecture for Scalable Content Analysis: Semi-automatic Annotation of User Generated Content . . . . .	158
<i>Mika Rautiainen, Arto Heikkinen, Jouni Sarvanko, and Mika Ylianttila</i>	
MashReduce – Server-Side Mashups for Mobile Devices . . . . .	168
<i>Joonas Salo, Timo Aaltonen, and Tommi Mikkonen</i>	
Open Service Platform for Pervasive Multimedia Services Development . . . . .	178
<i>Juho Perälä, Daniel Pakkala, and Juhani Laitakari</i>	

## Middleware

Gridification of a Radiotherapy Dose Computation Application with the XtremWeb-CH Environment .....	188
<i>Nabil Abdennhader, Mohamed Ben Belgacem, Raphaël Couturier, David Laiymani, Sébastien Miquée, Marko Niinimäki, and Marc Sauget</i>	
Leasing Service for Networks of Interactive Public Displays in Urban Spaces .....	198
<i>Marko Jurmu, Hannu Kukka, Simo Hosio, Jukka Riekkö, and Sasu Tarkoma</i>	
Yarta: A Middleware for Managing Mobile Social Ecosystems .....	209
<i>Alessandra Toninelli, Animesh Pathak, and Valérie Issarny</i>	
A Coordination Middleware for Orchestrating Heterogeneous Distributed Systems .....	221
<i>Nikolaos Georgantas, Mohammad Ashiqur Rahaman, Hamid Ameziani, Animesh Pathak, and Valérie Issarny</i>	

## Sensor Networks I

Wireless Sensor Networks Based on Publish/Subscribe Messaging Paradigms .....	233
<i>Hakan Cam, Ozgur Koray Sahingoz, and Ahmet Coskun Sonmez</i>	
Application-Centric Connectivity Restoration Algorithm for Wireless Sensor and Actor Networks .....	243
<i>Muhammad Imran, Abas Md. Said, Mohamed Younis, and Halabi Hasbullah</i>	
Link Quality-Based Channel Selection for Resource Constrained WSNs .....	254
<i>Markku Hänninen, Jukka Suhonen, Timo D. Hämäläinen, and Marko Hännikäinen</i>	

## Sensor Networks II

The Target Coverage Problem in Directional Sensor Networks with Rotatable Angles .....	264
<i>Chiu-Kuo Liang and Yen-Ting Chen</i>	

UBI-AMI: Real-Time Metering of Energy Consumption at Homes Using Multi-Hop IP-based Wireless Sensor Networks . . . . .	274
<i>Timo Ojala, Pauli Närhi, Teemu Leppänen, Jani Ylioja, Szymon Sasin, and Zach Shelby</i>	
An Accurate and Self-correcting Localization Algorithm for UWSN Using Anchor Nodes of Varying Communication Range . . . . .	285
<i>Manas Kumar Mishra, Neha Ojha, and M.M. Gore</i>	
<b>Author Index</b> . . . . .	295

# Applying Microsoft Research Technologies to the 4th Paradigm in Scientific Research

Daron G. Green

Microsoft Research,  
98052 Redmond , USA  
dagreen@microsoft.com

**Abstract.** In part as a recognition of the '4th Paradigm' in scientific discovery, in recent years Microsoft Research has steadily increased its interest in the application of computer science tools and technologies to breakthrough science. This has happened in diverse areas of research ranging from astronomy to oceanography, from molecular biology to 'big history' and from sociology to climatology. The emergence of cloud computing as a viable computing platform is extending our research capabilities and accelerating the rate at which the 4th Paradigm is becoming real for these various disciplines. We have enjoyed some significant successes by improving access to scientific information, for example with Microsoft Research's Worldwide Telescope, but have also discovered that there are many challenges remaining when one considers the relatively fragmented context within which most science is undertaken. This presentation explores the ways in which Microsoft is enabling changes in the way science is conducted, it will evidence how existing tools can be used and blended with new services, such as cloud computing, to improve the way in which data and information are discovered/shared/visualized and gives lessons learned from our collaborative research engagements associated with realizing our vision for the 4th Paradigm.

## Speaker Bio



Dr. Green is the General Manager of Microsoft Research Connection's and responsible for Microsoft Research's external engagement and investment strategy. His team and global portfolio includes diverse topics such as Health and Wellbeing, Education and Scholarly Communications, Computer Science and the Environment. Dr Green's initial research background was in molecular modeling and equations of state for fluid mixtures - his BSc is in Chemical Physics (1989, Sheffield) and Phd in molecular simulation of fluid mixtures (1992, Sheffield). He went on to undertake post-doctoral research in simulation of polymer and protein folding (1993-4, UCD). This naturally led to application porting and optimization for large-scale parallel and distributed computing in a range of application domains including computational chemistry (molecular dynamics

and quantum mechanical codes), radiography, Computational Fluid Dynamics and Finite Element analysis. Dr Green then moved more fully into HPC and was responsible for some of Europe's largest HPC Framework V programs for the European Commission, major HPC procurements in the UK for the UK Research Councils and UK Defense clients, he also led detailed investigations into the maturity and adoption for European HPC Software tools (published). From there Dr Green went to work for the SGI/Cray – helping to set up the European Professional Services organization from which he span out a small team out to establish the European Professional Services for Selectica Inc – Selectica specialized in on-line configuration/logic-engine technologies offered via web services. Given an HPC/distributed computing background and familiarity with the then embryonic area of Web Services, IBM invited Dr Green to help establish its Grid Computing Strategy and emerging business opportunity (Grid EBO) team. He subsequently moved to British Telecom to head-up its Global Services business incubation and, as part of this, in 2007 he established and launched BT's Sustainability practice - responsible for BT's business offerings to commercial customers which help reduce their carbon footprints and establish business practices which are sustainable in terms of their social and economic impact (published).



# Job Status Prediction – Catch Them Before They Fail

Igor Grudenic and Nikola Bogunovic

Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3  
10000 Zagreb, Croatia

{igor.grudenic, nikola.bogunovic}@fer.hr

**Abstract.** Jobs in a computer cluster have several exit statuses caused by application properties, user and scheduler behavior. In this paper we analyze importance of job statuses and potential use of their prediction prior to job execution. Method for prediction of failed jobs based on Bayesian classifier is proposed and accuracy of the method is analyzed on several workloads. This method is integrated to the EASY algorithm adapted to prioritize jobs that are likely to fail. System performance for both failed jobs and the entire workload is analyzed.

**Keywords:** Computer cluster, Job Status Prediction, Bayesian Classifier.

## 1 Introduction

Computer clusters are the most preferred distributed architecture type with a market share of 61% at the end of the 2008 [1]. Main reason for the popularity of computer clusters is its favorable price/computing performance ratio. Economics aside, computer clusters constitute 82% of the top 500 supercomputers [2], with the rest being massively parallel processors (MPP) and constellation systems.

Effective use of the computer cluster resources is enabled by the appropriate scheduler. Scheduling in computer cluster is performed whenever an event such as new job arrival, job completion or cancelation and resource malfunction occurs. Scheduling decision making rate must match the system event rate in order to avoid idling resources and to produce efficient schedules. Under the implicit time constraint scheduler must produce most feasible schedule in the uncertain environment.

Uncertainty of the environment is caused by unknown future jobs, future resource availability, runtimes and final statuses of the available jobs. Substantial work is done to statistically model future jobs [3] which resulted in improved average response time. Job runtimes have been also been a topic of exhaustive research. It is shown [4] that most of the users cannot predict job runtimes accurately. Half of the users are able to give more precise estimates in cases where underestimation doesn't cause early job termination, but overall accuracy improvement in such a scenario is not substantial. Numerous runtime prediction methods [5] [6] [7] [8] are proposed and it is concluded that simple [9] average runtime of the last two similar jobs is a very precise estimate.

Widely used EASY backfilling [10] algorithm was modified to accommodate runtime predictions [9] and a maximum of 28% reduce in average job slowdown is

reported. It is also noted that inaccuracies in runtime estimates can increase overall performance of the backfilling scheduler due to priority reversal that favors shorter jobs [11].

This paper deals with prediction of the job statuses, especially detection of the failed jobs, and elaborates potential use of these predictions. In Section 2 we define job statuses and give an analysis of resource consumption by each of the status classes. Prediction of job statuses that is based on the historical data of the several computer clusters is described in Section 3. Potential use of status prediction, namely prediction of the jobs that are likely to fail, that is implemented into the cluster job scheduler is presented in Section 4.

## 2 Job Statuses

Job status denotes a mode in which job exited the cluster system. Parallel Workloads Archive (PWA) [12] defines four job statuses: successfully completed job, failed job, job canceled by the user and job canceled by the system.

Failed jobs complete earlier than expected mostly due to programming error or exhaustion of reserved resources. Users can cancel jobs that are either running or waiting for execution. This usually happens when available intermediate results obtained from the job indicate that no longer execution is necessary and in cases of obvious configuration error. System can cancel jobs for many reasons, but it is typically done when job exceeds user requested runtime.

In the perfect computer cluster all the jobs would complete successfully, but this is seldom the case in real world scenarios. Distribution of job runtimes summarized by job statuses for ten computer clusters is presented in Fig. 1.

It can be observed that only HPC2N-2002 workload exclusively contains successfully completed jobs. User canceled jobs compose up to 8% of total runtime in five of the presented workloads. System canceled jobs make up 20% and 30% of total

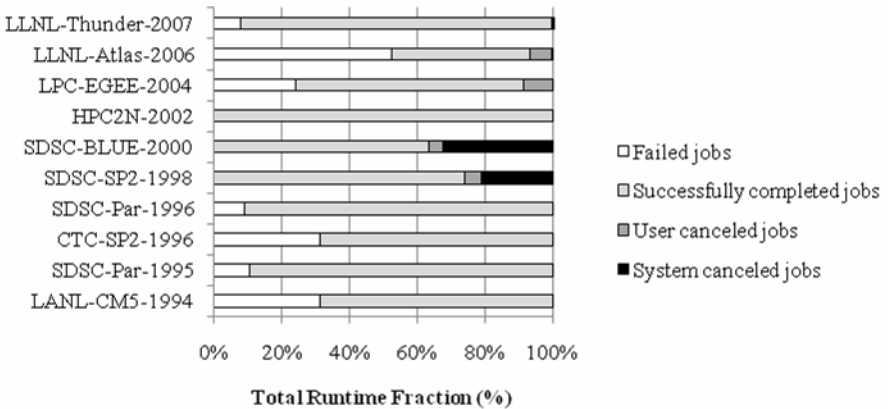


Fig. 1. Distribution of job runtimes grouped by statuses

runtime in workloads SDSC-BLUE-2000 and SDSC-SP2-1998. Failed jobs show up in seven computer clusters and are even using more than 50% of total runtime on the LLNL-Atlas-2006.

### 3 Job Status Prediction

Prediction of job statuses is possible by analyzing cluster usage history. Since it is impossible to achieve perfect status prediction rate, predicted job statuses should be used carefully in order to preserve expected behavior of the cluster scheduler.

There are several potential benefits that may arise from successful status prediction. If there is a high probability for a job to fail it is possible to raise its priority in order to force earlier execution. Earlier execution of a failed job would enable the user to deal with the malfunction sooner and to resubmit the job. Additionally users can recognize other similar jobs that may suffer from the same type of failure and cancel them prior to their execution. This could improve cluster efficiency since execution for the failed jobs may be prevented before or early in the execution. Main pitfall that may arise from prioritization of the jobs that are likely to fail is a possibility that a malicious user may misrepresent status of its own jobs trying to increase the chance of a better service for his future jobs.

Prediction of jobs that are canceled by the user can be used in job scheduling. Since users cancel jobs prior or during the execution it is possible to reduce the priority of jobs that will potentially be canceled. This would increase the chances for jobs to be deleted prior to their execution and could lead to better availability of the system. Problem with this approach is the absence of the user model and its decisions regarding canceling jobs which makes measurement of system improvement difficult.

Detection of jobs that are likely to be canceled by the system can be beneficial because jobs are usually canceled due to runtime underestimations. These jobs can be postponed to run at off-peak hours so their runtime can be prolonged without penalizing other jobs. Unfortunately this effect cannot be quantified using available workloads because realistic runtimes of canceled jobs are not known.

In this paper we focus on prediction of failed jobs since they can count for up to 50% of system runtime and effect of their prioritization can partly be measured. Jobs that are canceled by the user can be disregarded because they do not use significant amount of resources, while jobs canceled by the system have unknown real runtimes and benefits of their prediction cannot be determined.

Most basic technique of predicting the elements of a time series is a sliding window method. At discrete points in time (learning points) certain amount of workload history is used to build a classifier that is going to predict the future job statuses until the next learning point occurs as shown in Fig 2. The amount of history used for prediction and the number of jobs classified before the new classifier is built usually have 2:1 ratio.

The main classification issue is the choice of the sliding window size and the prediction method. Different methods for job status prediction and various sliding windows sizes are analyzed in section 3.1. Since optimal sliding window size is hard to determine statically the dynamic method for sliding window size computation is proposed in section 3.2. Accuracy of failed jobs prediction is given in section 3.3.

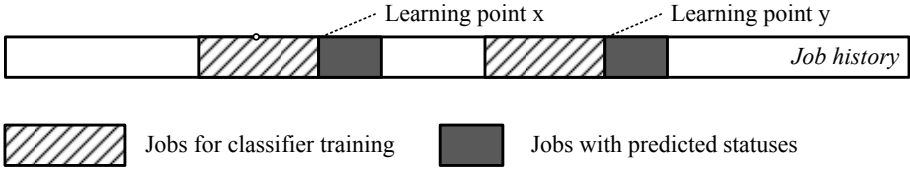


Fig. 2. Sliding window method

### 3.1 Prediction Method and Sliding Window Size

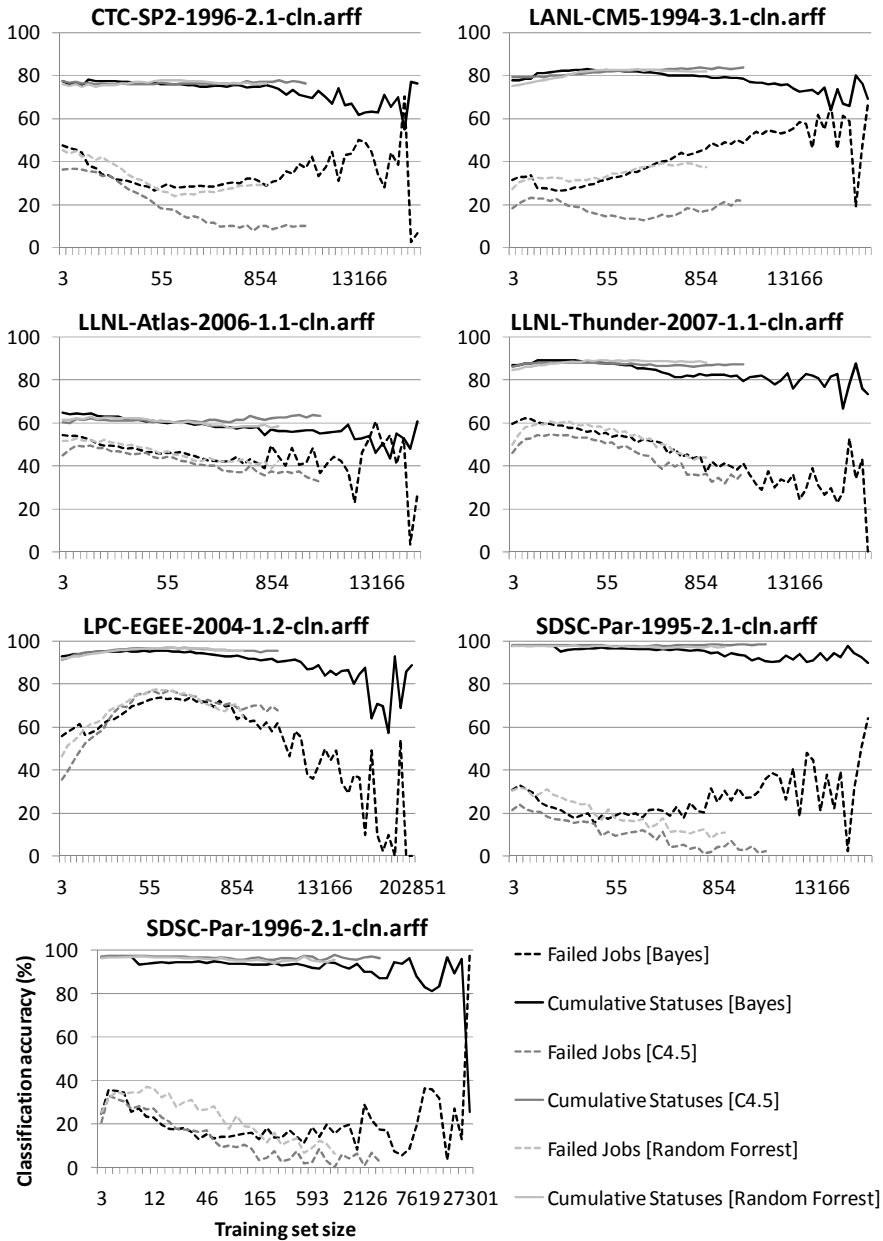
There are numerous classification methods that can be used to analyze data and perform predictions. In this paper we focused on three different classifier types: Naive Bayes Classifier [13], C4.5 classification tree [14] and random forests [15].

Data used for classifier training originate from PWA, but some of the job attributes are discarded because they do not describe job characteristics or are not known at the time the job enters the system. Attributes that are used include *Submit Time*, *Requested Number of Processors*, *Requested Time*, *Requested Memory*, *Status*, *User ID*, *Group ID*, *Executable (Application) Number*, *Queue Number* and *Partition Number*.

In order to determine sliding window size that corresponds to the classifier training set size we tested classification accuracy on a range of different sizes for the three classifiers. Classification accuracy results for seven computer clusters are presented in Fig 3. Results are presented for overall job status prediction and for prediction of failed jobs only. Sliding window method that is used is applied to the workload that is sorted by the *Submit Time* attribute and it is assumed that all the jobs in the classifier training set are already finished and their status is known. Although this doesn't hold in real life scenarios since order of job execution is dependent on the scheduling algorithm employed it can give a perspective on a behavior of different classifiers. In section 4 integration of failed job classification with EASY backfilling algorithm is analyzed and only finished jobs were used in classifier training.

Different training set sizes are represented on horizontal axis (in logarithmic scale) and prediction accuracy which represents true positives rate is given on vertical axis in Fig. 3. It can be observed that accuracy of three classification methods is very similar and best classifier among the three cannot be determined because it depends on analyzed computer cluster and size of the training set. Since Naive Bayes classifier has the lowest computational complexity and can be efficiently applied on a wide range of different training set sizes it is chosen as the most suitable for the detection of failed jobs.

Selection of the optimal training size that can be used invariantly of workload cannot be made since different sizes of job sets give optimal prediction results on different computer clusters. Even more, there are some irregularities in prediction rates that occur for larger training set sizes containing several thousands of jobs. Irregularities are even present for smaller training sets but these are smoothed out by the logarithmic scale and not visible on the charts.



**Fig. 3.** Comparison of prediction accuracy for three classifiers and multiple training set sizes

It is interesting to note that status prediction accuracy is high for very small training set sizes. In the next section we describe an algorithm for dynamic computation of feasible training set size that should be used to train job status classifier.

### 3.2 Dynamic Computation of Sliding Window Size

Algorithm for determining feasible classification training set size should be invariant to workload type and must be robust enough to avoid irregularities in prediction rates that can occur for similar training set sizes.

Dynamic training set size computation algorithm (DTSCA) is designed to run at fixed points in time and to determine best training size that could be achieved in the time period between last two calibration points as shown in Fig. 4. At *calibration point X* (time  $t_1$ ) calibration is performed in order to find the best training size to be used for building classifiers that are going to predict failed jobs in time period  $t_1-t_2$ . Calibration is performed on the jobs that finished execution between previous *calibration point X-1* and the current calibration point. Different train job sizes are tested for predictive accuracy and the best stable training size is used until the next calibration point. Time frame between two calibration points is set to two weeks.

Best stable training set size has a minimum of thousand jobs since smaller set sizes would call for more frequent classifier training that would hurt scheduler performance. Additionally, best stable training size is one that results in best accuracy when applied to the calibration job set and training sizes that are within 20% range of the picked size have no significant oscillations in predictive performance. Significant oscillations in predictive performance include prediction accuracies that are more than 25% lower than the ones acquired when using the best stable training size.

In order to perform calibration, job set that was used is limited to jobs executed from the previous calibration to the current calibration point. It is possible to use the entire known workload to perform the calibration but this showed no improvement on prediction accuracy and consumed more resources when computed. Although we performed entire calibration at a calibration point this computation can be performed incrementally as jobs in the system complete execution.

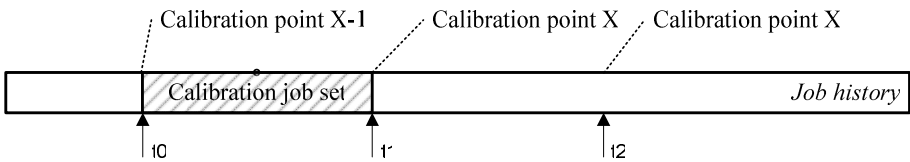


Fig. 4. Dynamic computation of training set size

### 3.3 Failed Job Prediction Results

For prediction of failed jobs three different prediction methods are tested. First one is a simple heuristic method obtained from detailed insight into data, the second one is based on Bayesian classifier with dynamic computation of training window size and the third is a hybrid method composed of both simple heuristic and a Bayesian based classifier.

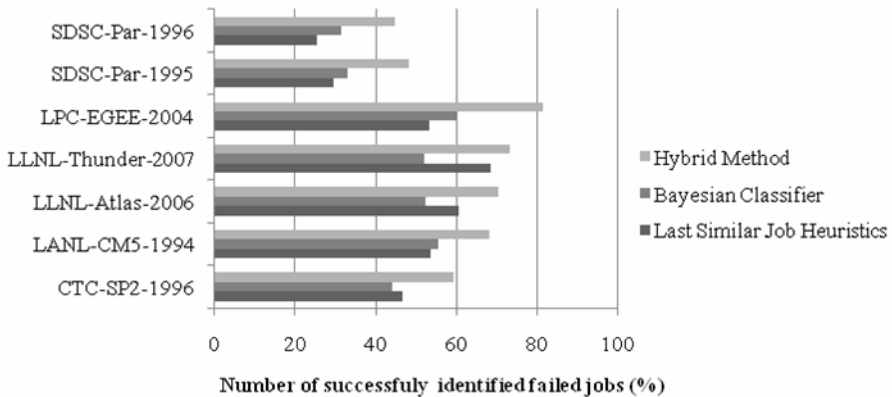
Since it is noted that classifiers built using small training set sizes have very high prediction accuracy, detailed analysis of the data is performed and it is concluded that it is very likely for a user to submit series of failing jobs to the computer cluster.

Using this observation simple ‘last similar job’ heuristics is constructed that predicts job to fail if the previous similar job completed with failure. Similar job is defined as a job from the same user and of the same application type (*Executable Number* attribute).

Hybrid method is defined as a voting system between simple ‘last similar job’ heuristics and Bayesian classifier and it predicts job to fail if either of the two classifiers guess the target job will most likely fail. Probability of failure in hybrid method is equal to 100% if predicted according to the last similar job. In cases where prediction of failure comes exclusively from the Bayesian classifier appropriate probability is used.

This aggressive approach is employed in order to capture as much failing jobs as possible. Detection of other jobs statuses in a hybrid method is done exclusively by the Bayesian classifier.

Comparison of prediction accuracy for the three prediction methods is presented in Fig. 5. It can be easily observed that hybrid method outperforms both of the other prediction methods as expected. Bayesian classifier is a better predictor than ‘last similar job’ heuristics for four of the presented workloads. The best prediction rate of 81.36% is achieved for detection of failed jobs on a LPC-EGEE-2004 workload. For some workloads detection of failed jobs is less successful but is always manages to select at least 45% of jobs that are about to fail.



**Fig. 5.** Failed job prediction results

Level of prediction overlapping for Bayesian classifier and ‘last similar job’ heuristics can be deduced by examining the difference between hybrid method and best of the other two methods. The difference ranges from 4,69% to 21,05% which indicates that for some computer workloads like LLNL-Thunder-2007 where overlapping is high one of the other methods can be used instead of hybrid method.

True positives rate that indicates fraction of successfully classified jobs is just one of the measures for classifier quality. Better insight into the predictive properties of the classifier can be depicted from the lift chart. Lift chart for the hybrid method of prediction applied on CTC-SP2-1996 workload is given in Fig. 6. It is visible from

the chart that hybrid method outperforms random choice. For example, if 20% of the total job population is picked up randomly it is likely for that pick to contain 20% of all the failed jobs. If 20% of the jobs with highest probability according to hybrid method are chosen then the choice would contain 50% of the jobs that eventually fail.

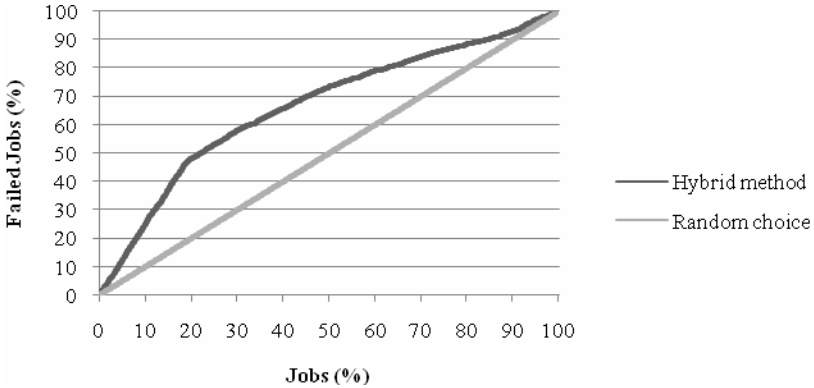


Fig. 6. Comparison of hybrid method for failed jobs detection and random choice

#### 4 Job Scheduling Using Failed Job Predictions

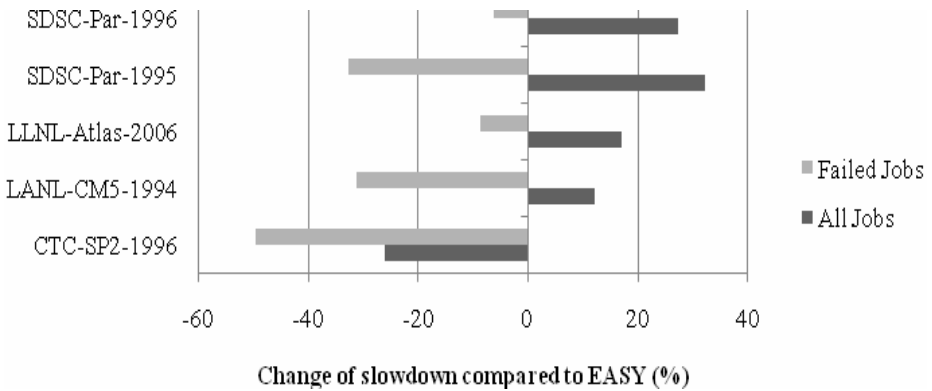
Job scheduling in computer cluster assigns jobs that have different priorities to available resources. There are many algorithms that deal with different types of jobs. In this paper we use modified EASY backfilling algorithm in order to accommodate job status predictions. Every iteration of the EASY algorithm makes reservation for the highest priority job while other jobs may be scheduled to fill ‘holes’ in the schedule respecting the reservation made. Order in which jobs are scanned to fill the ‘holes’ is done by decreasing job priority. EASY algorithm ensures high utilization of the resources while implicitly respecting priorities, although priority inversion is possible and depends on available jobs and current resource assignment. Starvation of long or resource intensive jobs is prevented by EASY but the level of service for these jobs is usually lower than average.

In order to integrate failed job prediction with EASY scheduler we changed the jobs priorities scheme to follow the likelihood of jobs to fail. Jobs that are most likely to fail are given higher priority with the exception of the highest priority job. Highest priority job in the new scheme is left the same in order to prevent starvation. Prioritization of the jobs that are likely to end in an error is done to provide failure information to the users as quickly as possible which enables them to fix the error sooner and opens the possibility for the user to cancel other similar jobs that might also fail.

Since no model of user behavior in this scenario exists we measured quality of the system response for failed jobs and all the jobs in the system. Quality of response for five workloads expressed as a change in average slowdown compared to traditional EASY is given in Fig. 7. Slowdown is a ratio of time job spent in the system and the execution time of the job.



It can be observed that for all the simulated workloads slowdown of failed jobs decreased from 7% up to 50%. Improved failed jobs slowdown caused the slowdown of all the jobs in the system to increase up to 32% for all except the CTC-SP2-1996 workload. This was expected since jobs that are determined as likely to fail by the hybrid method consume up to 50% more time and resources than average job in the system. Anomaly that is shown for CTC-SP2-1996 workload in which improving priority of failed jobs that are longer and harder on resources leads to 25% decrease in overall system slowdown can be explained by better packing of the schedule. This can be caused by the similarities in jobs that are likely to fail and raising priorities of similar jobs can lead to better resource allocation.



**Fig. 7.** Comparison of EASY with prioritization of failed jobs and traditional EASY algorithm

## 5 Conclusion

In this paper we analyzed different jobs statuses and potential benefit that may arise if those are known in advance. Jobs that fail are recognized to be resource intensive but usefulness of their execution is not clear. It is assumed that some advantage may be gained with raising priority of failing jobs since users might favor failure information to come early as possible. Additionally users may cancel similar queued jobs that can also be erroneous and unload the system.

We designed a hybrid method for failed job prediction based on Bayesian classifier and simple ‘last similar job’ heuristics. It is shown that up to 80% of failed jobs can be predicted before they begin execution. Hybrid prediction method is integrated into the EASY scheduler with a goal to increase the priority of the failed jobs. This resulted in up to 50% decrease in failed job slowdown, but overall system performance suffered up to 32% slowdown increase. Such a behavior was expected since failed jobs use up to 50% more runtime and resources then the average system job. Direct benefits of failed jobs prioritization could not be measured due to lack of data and non existing user model. In the future we intend to measure user satisfaction with earlier failed job detection, as well as the way user behave on this information.

## References

1. IDC HPC Market Update, [http://www.hpcadvisorycouncil.com/events/china\\_workshop/pdf/6\\_IDC.pdf](http://www.hpcadvisorycouncil.com/events/china_workshop/pdf/6_IDC.pdf)
2. TOP500 Supercomputing Sites, <http://www.top500.org/>
3. Barsanti, L., Sodan, A.: Adaptive Job Scheduling via Predictive Job Resource Allocation. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2006. LNCS, vol. 4376, pp. 115–140. Springer, Heidelberg (2007)
4. Bailey Lee, C., Schwartzman, Y., Hardy, J., Snavely, A.: Are user runtime estimates inherently inaccurate? In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 253–263. Springer, Heidelberg (2005)
5. Gibbons, R.: A Historical Application Profiler for Use by Parallel Schedulers. In: Feitelson, D., Rudolph, L. (eds.) IPPS-WS 1997 and JSSPP 1997. LNCS, vol. 1291, pp. 58–77. Springer, Heidelberg (1997)
6. Smith, W., Foster, I., Taylor, V.: Predicting Application Run Times Using Historical Information. In: Feitelson, D., Rudolph, L. (eds.) IPPS-WS 1998, SPDP-WS 1998, and JSSPP 1998. LNCS, vol. 1459, pp. 122–142. Springer, Heidelberg (1998)
7. Krishnaswamy, S., Loke, S.W., Zaslavsky, A.: Estimating computation times of data-intensive applications. *IEEE Distributed Systems Online* 5(4) (2004)
8. Kapadia, N.H., Fortes, J.A.B., Brodley, C.E.: Predictive Application-Performance Modeling in a Computational Grid Environment. In: the Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing, pp. 47–54 (1999)
9. Tsafirir, D., Etsion, Y., Feitelson, D.G.: Backfilling Using System-Generated Predictions Rather than User Runtime Estimates. *IEEE Transactions on Parallel and Distributed Systems* 18(6), 789–803 (2007)
10. Lifka, D.A.: The ANL IBM SP Scheduling System. In: Feitelson, D., Rudolph, L. (eds.) IPPS-WS 1995 and JSSPP 1995. LNCS, vol. 949, pp. 295–303. Springer, Heidelberg (1995)
11. Tsafirir, D., Feitelson, D.G.: The Dynamics of Backfilling: Solving the Mystery of Why Increased Inaccuracy May Help. In: Proceedings of 2006 IEEE International Symposium on Workload Characterization, pp. 131–141 (2006)
12. Parallel Workloads Archive, <http://www.cs.huji.ac.il/labs/parallel/workload/>
13. Han, J., Kamber, M., Pei, J.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco (2005)
14. Quinlan, J.R.: *C4.5: Programs for machine learning*. Morgan Kaufmann, San Francisco (1993)
15. Breiman: Random Forests. *Machine Learning* 45, 5–32 (2001)

# Group-Based Gossip Multicast Protocol for Efficient and Fault Tolerant Message Dissemination in Clouds<sup>\*</sup>

JongBeom Lim, JongHyuk Lee, SungHo Chin, and HeonChang Yu<sup>\*\*</sup>

Dept. of Computer Science Education, Korea University  
{jblim, spurt, wingtop, yuhc}@korea.ac.kr

**Abstract.** Cloud computing is an Internet-based computing paradigm that provides services in a virtualized form composed of plenty of sharable resources. In cloud computing environments, gossip protocols are engaged as a method to rapidly disseminate the state information for innumerable resources. Although the gossip protocols provide a robust and scalable multicast, there is a drawback that requires redundant messages in satisfying 100% of reliability. In our study, we propose a Group-based Gossip Multicast Protocol to reduce the message overhead while delivering the state information efficiently and fault tolerantly. Furthermore, we verified the performance of the proposed protocol through experiments.

**Keywords:** Gossip Protocols, Cloud Computing, Multicast.

## 1 Introduction

Many distributed systems can benefit from a reliable application-level multicast because such large scale and dynamic systems are made up of inter-domain regions, which are difficult to handle and require high maintenance cost to an IP-level multicast protocol.

As an application-level multicast, Gossip multicast protocols are recognized as a useful building block for dealing with environments in which a system has the highly dynamic properties of nodes and links [2]. In this protocol, each node periodically contacts its neighbors that are selected at random through the peer sampling service [3]. With gossip multicast protocol, several valuable services can be provided such as dissemination, aggregation and topology management in distributed systems.

Cloud computing is an emerging information technology for delivering computing resources such as networks, servers, storage, or software applications that are built on virtualization technologies [1]. One characteristic of cloud computing is elasticity. In other words, because the computing environments are based on loosely coupled architecture, computing resources are easily added to and removed from the system.

---

<sup>\*</sup> This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (20100023947).

<sup>\*\*</sup> Corresponding author.

Likewise, gossip multicast protocols have the similar characteristics that individual nodes constituting an overlay network can join and leave at any time.

Although gossip protocols provide a robust and scalable multicast, there is a drawback that requires redundant messages in satisfying 100% of reliability. Another potential problem of gossip multicast protocols is that can be suffered from failed nodes. As the number of failed nodes increase, time to achieve some degree of reliability is also delayed.

To reduce the message overhead while delivering the messages efficiently and fault tolerantly, we present a Group-based Gossip Multicast Protocol. Our proposed algorithms are composed of two parts: (a) Self-organization that builds overlay networks, partial views and group views; (b) Message dissemination to diffuse the messages throughout the system by using view information constructed by the Self-organization phase.

The rest of this paper is organized as follows. Section 2 describes existing gossip multicast protocols, while our proposed algorithms are explained in section 3. In section 4, experimental results are presented. Finally, we conclude the paper in section 5.

## 2 Related Work

The literature concerning the overlay construction and message dissemination using gossip protocols is extensive. In this section, we briefly present several works closely related to ours.

Scamp [4] is a probabilistic scalable membership protocol that does not depends on global membership information, which is unsuitable in large-scale distributed systems. Instead, in Scamp each node has the knowledge of partial neighbor information that is called *apartial view*. The size of a partial view in this protocol is  $(c+1)\log(N)$ , where  $N$  is the total number of nodes in the system and  $c$  is a design parameter. In this manner, Scamp can solve the scalability problem.

Hiscamp [5], a self-organizing hierarchical membership protocol, implements two levels of gossiping with Scamp. In other words, nodes in the system are grouped into cluster measured by locality, and each node can interact with nodes within its cluster or in other clusters. Hence, each node should maintain two partial views: *iview* and *hview*. However, Hiscamp has a single point of failure due to its hierarchical structure, thus must take defensive measures.

Cyclon [6] introduces a view-changing protocol called a *shuffle*. With a shuffle operation, each node can exchange their partial view information when gossiping by comparing the freshness. It assumes a node that has the old timestamp more likely to be failed or has leaved the system. Thus, each node tends to have alive nodes information in the presence of failure or churn.

Hyparview [7] proposed a reliable membership protocol that ensures high degree of reliability in spite of a numerous number of failed nodes. This characteristic of fault tolerance is attributed to have two distinct partial views, namely an *active view* and a *passive view* on each node. When gossiping, each node maintains their passive view for the backup purpose. If a node notices that any of its neighbors in an active view are failed, then an active view is reconstructed by using information of a passive view.

Clon [8] aimed at providing overlay construction and message dissemination expanding gossip protocols to cloud computing environments. Clon exploits the locality to reduce the load imposed on long-distance links. However, their experiments are performed assuming that there are 5 local areas; but this assumption is not suited for dynamic and large-scale distributed systems.

One of inherent properties of a gossip protocol is that of redundancy that makes it resilient in the occurrence of failure. Redundancy of a gossip protocol, however, sometimes causes overhead costs on communication links. To reduce redundant messages of a gossip protocol, we introduce a Group-based Gossip Multicast Protocol that allows reaching some degree of reliability efficiently in a fault tolerant manner.

### 3 Group-Based Gossip Algorithms

In a flat gossip protocol, each node has node information fields with Node ID, which is a unique number in the system, and Timestamp to represent the time a node is created. In our proposed protocol, on the other hand, the additional information fields are provided, that is, *Group ID* and *Group Size* described in Figure 1. These node information fields are used to build overlay network and disseminate messages over the network links.

<b>Node ID</b>	<b>Timestamp</b>	<b>Group ID</b>	<b>Group Size</b>
----------------	------------------	-----------------	-------------------

**Fig. 1.** Node information fields in a group-based gossip multicast protocol

The group-based gossip algorithm is divided into two parts: (a) a self-organization algorithm; (b) a message dissemination algorithm. In first part, a self-organization algorithm, *partialView* and *groupView* are generated based on system parameters such as the size of *partialView(k)*, the number of group (*groupCount*) and group member (*groupSize*).

Figure 2 shows the pseudo-code of the self-organization algorithm, which is executed once before the message dissemination algorithm is performed. At the beginning, the self-organization algorithm fetches the values of *groupCount*, *groupSize* and *partial ViewSize* (lines 2-4 in Fig. 2) and these values are determined by system configuration. As doing in the flat gossip protocol, a *partialView* is filled with the list of Node ID selected at random for every node in the system (lines 5-9 in Fig. 2).

With values of *groupCount* and *groupSize*, a *groupView* and a *groupProperty* are maintained (lines 10-28 in Fig. 2) and each node has at most one of *groupIDs*. During constructing a *groupView*, if a node selected by the peer sampling service at random, already has a *groupID*, then the peer selection phase is repeated (lines 12-15 in Fig. 2). To avoid duplicated node information in a *groupView*, because gossip protocols use a random approach in choosing a node, the *checkDuplicated* function is invoked with *groupList* (lines 17-20 in Fig. 2). After *groupList* are made properly, group information including *groupList*, *groupID* and *groupSize* are assigned to nodes in a *groupList* (lines 22-25 in Fig. 2). Finally, we clear a *groupList* and set the group representative of a group to one of nodes in a group. Given this, a group representative can interact with its group members to disseminate the messages, which is described later on this section.

```

1: do Self-organization
2:   groupCount ← getGroupCount()
3:   groupSize ← getGroupSize()
4:   partialViewSize ← getPartialViewSize()
5:   for each n ∈ System
6:     for each partialView ∈ n
7:       partialView ← getPeer()
8:     end for
9:   end for
10:  for i=0; i < groupCount; i++
11:    for j=0; j < groupSize; j++
12:      peer ← getPeer()
13:      if peerGroupID != ∅ then
14:        continue
15:      end if
16:      call addGroupList(peer)
17:      if checkDuplicate(groupList) then
18:        call removeGroupList(peer)
19:        continue
20:      end if
21:    end for
22:    for each n ∈ groupList
23:      call setGroupView(groupList)
24:      call setGroupProperty(groupId, groupSize)
25:    end for
26:    call setGroupRepresentative()
27:    call clearGroupList()
28:  end for
29: end do

```

Fig. 2. Self-Organization Algorithm

The pseudo-code of the message dissemination algorithm is depicted in Figure 3 and has an additional form of the flat dissemination algorithm by adding the dissemination phase to a group. Unlike the self-organization algorithm, the message dissemination algorithm is executed on all nodes in the system. If a node itself is a group representative, the node disseminates the messages to all the members in a groupView (lines 2-6 in Fig. 3). Otherwise, each node contacts to its neighbors in a partialView. Because of this procedure, the occurrence of the duplicated messages delivery within a group is reduced. However, in the case of fault scenario, the equality check (line 2 in Fig. 3) is ignored to avoid circumstances where a group representative is failed resulting in the absence of group communications.

When gossiping with nodes in a partialView, to increase the probability of contact with other group members, another equality check is performed (lines 8-13 in Fig. 3). In other words, if a node belongs to a particular group, the node will not select a neighbor that has the same groupId with its own, hence duplicate messages are can be further reduced. Afterward, the node sends a message to its neighbors (line 14 in Fig. 3). In this step, we can choose one of the three ways to communicate with others: Push mode, Pull mode, and Push-pull mode.

```

1: do Dissemination
2:   if node == groupRepresentative then
3:     for each n ∈ groupView
4:       SendMessages(n, message, myself)
5:     end for
6:   end if
7:   peer ← n ∈ partialView
8:   while myGroupID == peerGroupID
9:     peer ← n ∈ partialView
10:    if myGroupID == peer.GroupID then
11:      continue
12:    end if
13:  end while
14:  SendMessages(peer, message, myself)
15: end do

```

Fig. 3. Message Dissemination Algorithm

## 4 Performance Evaluation

To evaluate the group-based gossip multicast protocol, we use the Peersim [9] simulator that supports extreme scalability and dynamicity. System parameters and their values used for the group-based gossip protocol are listed in Table 1. At the initialization step, a unique identifier is assigned to every node and there is one node that has the message to disseminate in the system.

Table 1. Evaluation parameters and settings for the group-based gossip multicast protocol

Parameters	Value
N (Total Number of Nodes )	50,000
k (Size of PartialView)	20
Cycle	20
Fanout	1
G (Total Number of Groups)	20, 40, 60, 80
M (The Number of Members in a Group)	100, 200, 300, 400

Moreover,  $G$  (total number of groups) and  $M$  (the number of members in a group) parameters are used to test the group-based gossip multicast protocol. According to these two parameters, the number of nodes that did not receive the message is measured comparing with the flat gossip multicast protocol using the push-pull mode.

### 4.1 Dissemination

Figure 4 shows the number of nodes that did not receive the message depending on group sizes ( $G$ ) and the number of members in a group ( $M$ ) as the cycle progresses when the total number of nodes is 50,000. In case of  $M = 100$  (Fig. 4(a)), the required numbers of cycle sare 7, 6, 6, and 6 when the number of groups is 20, 40, 60 and 80, respectively, to disseminate the message to all the members in the system. This means

that even if only 4% of nodes are belonging to a group, there are performance benefits compared to the flat gossip protocol, which requires 9 cycles to spread the message to all members.

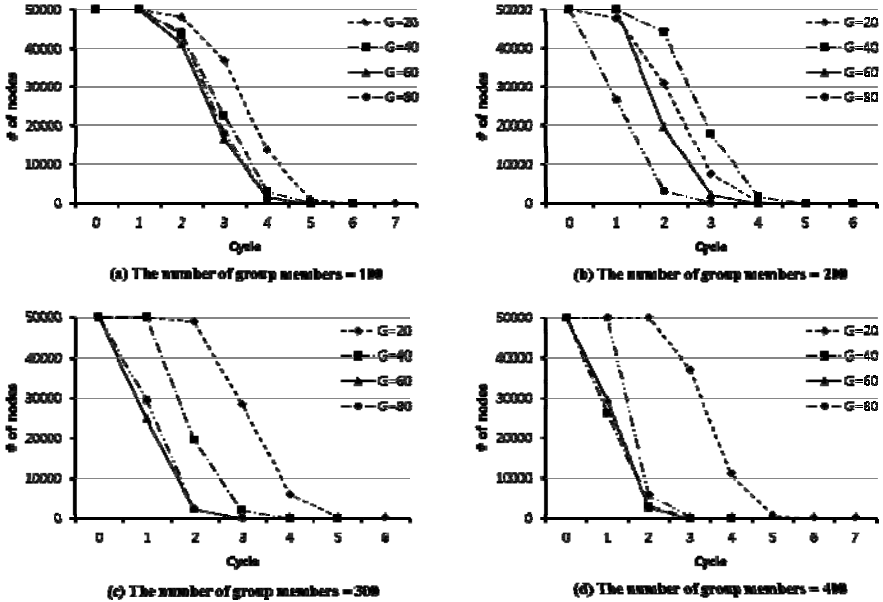


Fig. 4. The numbers of nodes that did not receive a message according to group sizes ( $G$ ) and group members ( $M$ )

Furthermore, as the number of groups and group members are increased, we have confirmed that the required number of cycle to disseminate the message to all nodes is decreased. Especially, if  $M = 400$ , the required numbers of cycles are 7, 4, 4 and 4 when the number of groups is increased from 20 to 80 by increments of 20. In other words, when the number of group members is 400 and the number of a group is 40, there is a performance improvement of about 55% in terms of the required number of cycles compared to the flat gossip protocol.

On the other hand, the number of message payloads of the group-based gossip multicast protocol on each cycle is  $N + (|Group| \times |\{n: g_i \in Group\}| - |Group|)$ , where  $N$  is the total number of nodes in the system and  $g_i$  is the number of group members in a group while the flat gossip multicast protocol generates  $N$  message payloads on a cycle.

Figure 5(a) shows the numbers of cycles required to achieve 100% of reliability and are 5, 3, 4, 4 and 4, respectively, when *fanout* is 2 and 3 in the flat gossip protocol; the number of groups is 40, 60 and 80 in the group-based gossip protocol provided that the number of group members is 400. Again, the number of message payloads is 100000, 150000, 65960, 74940, and 91920, respectively. In other words, the experiments show that the group-based gossip protocol requires fewer numbers of cycles and message payloads than the flat gossip protocol.



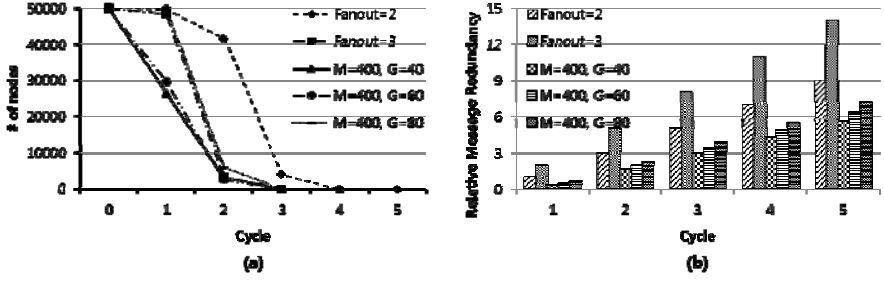


Fig. 5. Comparison of the numbers of nodes did not receive a message (a), and of Cumulative RMR value (b) between the flat gossip protocol and the group-based gossip protocol

Additionally, to measure redundant messages generated for each protocol, the *RMR* (Relative Message Redundancy) [10] metric is used and its definition is as follows:

$$RMR = \left( \frac{M}{N-1} \right) - 1$$

Where  $N$  is the total number of nodes in the system and  $M$  is the number of message payloads. The higher *RMR* value means that more redundant messages are produced.

Figure 5(b) shows the cumulative *RMR* values of the flat gossip protocol (*fanout=2*, *fanout=3*) and the group-based gossip protocol ( $G=40$ ,  $G=60$ ,  $G=80$ ) provided that the number of group members is 400. The cumulative *RMR* values are gradually increased as the number of cycles increase. Especially, to satisfy 100% of reliability, the cumulative *RMR* value of the flat gossip protocol (*fanout=3*) is about 8 (at cycle 3) and that of the group-based gossip protocol ( $G=40$ ) is about 4.27 (at cycle 4). This means the group-based gossip protocol has fewer *RMR* value by about 46% compared to the flat gossip protocol.

## 4.2 Fault Tolerance

In cloud computing environments, there are a lot of nodes having resources and the nodes can be failed and be removed for administrative or catastrophic reasons. To this end, we have evaluated the reliability of two protocols: the flat gossip protocol and the group-based gossip protocol. We also consider following two scenarios for the experiments:

1. Nodes are failing at a constant rate at each cycle: We assume that all nodes have a constant failure rate. Hence, some numbers of nodes are failing at each cycle based on a failure rate. Reliability of protocols is measured according to failure rates that are 1%, 2%, 3%, 4% and 5%.
2. Nodes are failing at a constant rate at one cycle: Because of administrative or disastrous reasons, failures are occurred at one cycle after one gossip exchange takes place. In other words, reliability is measured according to failure rates from 10% to 90% by increments of 10%. Unlike scenario 1 above, once failures are occurred at the cycle, no further failures are happened.

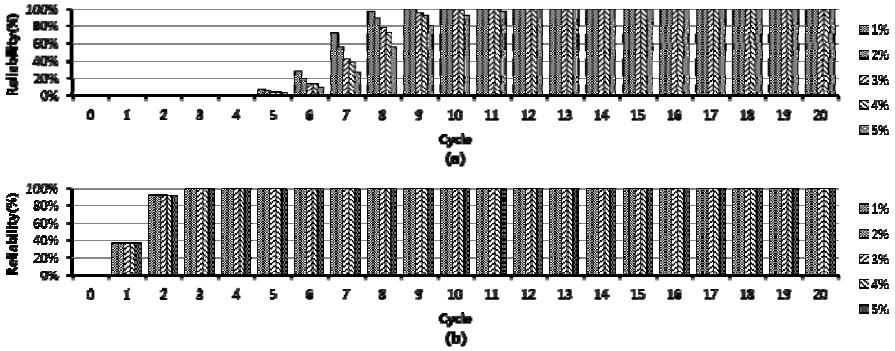


Fig. 6. Reliability of the flat gossip protocol (a) and the group-based gossip protocol ( $M=400, G=80$ ) (b) for scenario 1

Furthermore, to cope with the situations where failures are occurred, the message dissemination algorithm is configured to communicate with its all group members on every node belonging to a group as explained in section 3. By applying this approach, the situations where group communications do not happen although a node belongs to a group when a group representative is failed can be avoided. Consequently, our group-based gossip multicast protocol does not require an overlay reconstruction phase and its overhead costs.

Figure 6 shows the reliability of the flat gossip protocol and the group-based gossip protocol for scenario 1. As shown in Figure 6(a), in case of the flat gossip protocol with 1% of failure rate the required number of cycles is 9 in satisfying 99.5% of reliability. In addition, if a failure rate is 5%, the required number of cycles is 14 in satisfying 99.5% of reliability.

On the other hand, in Figure 6(b), when the group-based gossip protocol ( $M=400, G=80$ ) is used the required numbers of cycles are 3, 3, 3, 3 and 4 in satisfying 99.5% of reliability depending on the failure rates of 1%, 2%, 3%, 4% and 5%, respectively. When a failure rate is 5%, the group-based gossip protocol requires fewer cycles by about 71% than that of the flat gossip protocol in satisfying 99.5% of reliability.

In Figure 7, the reliability of the flat gossip protocol and the group-based gossip protocol for scenario 2 is presented. If a failure rate is 10%, in the flat gossip protocol, the required number of cycles is 10. When failure rate is 70%, however, reliability is below 95% even though when the number of cycle is 20 in the flat gossip protocol. If failure rates are 80% and 90%, reliability of the flat gossip protocol is 17.56% and 0.40%, respectively.

Finally, Figure 7(b) depicts the reliability of the group-based gossip protocol ( $M=400, G=80$ ). When a failure rate is 10%, the required number of cycles in satisfying 99.5% of reliability is 4, which are fewer cycles by about 60% than that of the flat gossip protocol. Furthermore, if failure rates are 80% and 90%, reliability of the group-based gossip protocol is 99.8% and 96.3%. This means that the performance differences of reliability is approximately 82% and 90%, respectively, between the two protocols.

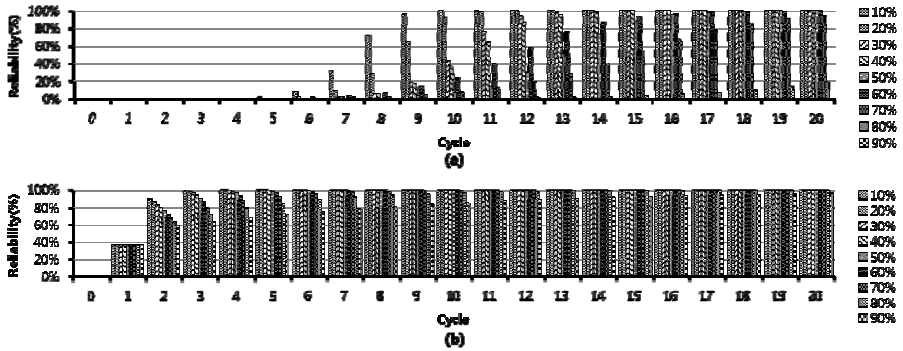


Fig. 7. Reliability of the flat gossip protocol (a) and the group-based gossip protocol ( $M=400$ ,  $G=80$ ) (b) for scenario 2

## 5 Conclusions

The nature of cloud computing is that service providers have to offer rapid provisioning of virtual resources according to SLAs (Service level agreements). As cloud computing environments have the dynamic and large-scale characteristics, it is crucial to disseminate information of nodes as fast as to reduce the probability of provisioning with failed nodes in order to satisfy SLAs requirements.

In this paper, we introduced the group-based gossip protocol to reduce message overhead and to tolerate failures of nodes in cloud computing environments. With the group-based gossip protocol, propagation delays can be decreased in satisfying some degree of reliability; atomic message deliveries can be achieved efficiently if failures do not occur. Experiments show that our proposed protocol is superior to the flat gossip protocol with respect to reliability and the required number of cycles to disseminate messages without requiring an overlay reconstruction phase.

## References

1. Buyya, R., Yeo, C.S., Venugopal, S.: Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. IEEE Computer Soc., Los Alamitos (2008)
2. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Voshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. SIGOPS Oper. Syst. Rev. 41, 205–220 (2007)
3. Jelasity, M., Guerraoui, R., Kermarrec, A.-M., van Steen, M.: The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 79–98. Springer, Heidelberg (2004)
4. Ganesh, A.J., Kermarrec, A.-M., Massoulié, L.: SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication. In: Crowcroft, J., Hofmann, M. (eds.) NGC 2001. LNCS, vol. 2233, p. 44. Springer, Heidelberg (2001)

5. Ganesh, A.J., Kermarrec, A.-M., Massoulié, L.: HiScamp: self-organizing hierarchical membership protocol. In: Proceedings of the 10th Workshop on ACM SIGOPS European Workshop. ACM, Saint-Emilion (2002)
6. Voulgaris, S., Gavidia, D., van Steen, M.: CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. *Journal of Network and Systems Management* 13, 197–217 (2005)
7. Leitao, J., Pereira, J., Rodrigues, L.: HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, pp. 419–429 (2007)
8. Matos, M., Sousa, A., Pereira, J., Oliveira, R., Deliot, E., Murray, P.: CLON: Overlay Networks and Gossip Protocols for Cloud Environments. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009. LNCS, vol. 5870, pp. 549–566. Springer, Heidelberg (2009)
9. The PeerSim Simulator, <http://peersim.sourceforge.net>
10. Leitao, J., Pereira, J., Rodrigues, L.: Epidemic Broadcast Trees. In: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems. IEEE Computer Society, Los Alamitos (2007)

# Co-management of Power and Performance in Virtualized Distributed Environments

Mohsen Sharifi, Mahsa Najafzadeh, and Hadi Salimi

Distributed Systems Laboratory,  
School of Computer Engineering,  
Iran University of Science and Technology,  
Tehran, Iran  
{msharifi,hsalimi}@iust.ac.ir,  
mahsa\_najafzadeh@comp.iust.ac.ir

**Abstract.** Rapid growth of large-scale applications and their widespread use in research and industry has led to dramatic increases in energy consumption in enterprise data centers and large-scale distributed systems such as Grids. Any attempt at reducing the energy consumption without concern for performance can be destructive and deteriorate the overall efficiency of data centers and large-scale distributed systems running such applications. In this paper, we present an optimization model for resource management in virtualized distributed systems to minimize power costs automatically while satisfying performance constraints. The objective of our model is to keep the utilization of servers near to an optimum point to prevent performance degradation. The model includes two objective functions, one for power costs and another for performance. Using the objective functions, we present a scheduling algorithm to place a set of virtual machines on a set of servers dynamically so that to integrate power management with performance management. We show experimentally that the proposed scheduler consumes approximately 24% less energy than static power management techniques while maintaining comparable performance.

**Keywords:** power management, performance, virtualization technology, consolidation.

## 1 Introduction

Research and industrial communities' interest in large-scale complex computing systems has resulted in the development of special infrastructures for distributed systems such as Grids. Consumption of energy grows dramatically in these infrastructures with the high volumes of demands and enormous amounts of computing and storage operations. The increased energy consumption, particularly in large-scale distributed environments such as in computational Grids, has raised many challenges including higher carbon footprints, cooling costs, degradation of performance, and reliability issues. Therefore, energy efficiency and cost reduction, as QoS parameters, have been in the center of attention of developers and users of large-scale distributed systems. Some power management techniques such as energy efficient hardware, for example

DVFS-enabled CPUs [1], and power-aware algorithms have been utilized to improve energy consumption. In addition to these policies, exploitation of virtualization techniques [2] has considerably improved energy efficiency of data centers and optimizes resource utilization.

Virtualization technology (VT) is a powerful technology that has been utilized in many computing systems ranging from mobile devices to systems with high processing power. Virtualization allows the consolidation of a set of virtual machines (VMs) to a single physical server. Servers in idle state, for example at 5% processor utilization or lower, consume almost equivalent energy to a server at 70% processor utilization. Nonetheless, consolidation reduces the number of idle servers by the capability to run more than one VM on a server and so can help to save energy. When consolidation is implemented properly, it can improve energy efficiency. Nevertheless, its incorrect implementation can be more detrimental to the performance and offset the energy-saving gaining.

Several existing consolidation models [3,4,6] have merely focused on the maximization of resource utilization, and as a result have minimized the number of active servers. Hence, the level of complexity and application's performance degradation increases in the remaining servers even though the number of active servers may decrease by this method. In addition, attempts to maintain servers in high utilization levels, from 95% to 100%, are not reasonable due to the possible system crashes in the face of spikes in demands and failures. Therefore, it is essential to develop new efficient methods to reconcile power and performance management in data centers.

In contrast to existing related works, we propose a smart consolidation approach for energy efficiency by taking into account both performance and energy costs. For this purpose, we present an optimization model that includes two objective functions, one for power cost and another for the performance. In addition, we cope with the cost of VM migration and turning servers on or off in our consolidation model. Using the objective functions, we present a scheduling algorithm for energy-aware allocation of VMs to physical servers. The optimization problem is a NP-hard problem that has remained unsolved for a long time. To address this issue, we use a simulated annealing (SA) method [14], which significantly reduces the search space and planning time in large scales.

We have organized the rest of paper as follows. Section 2 discusses some notable related works. Sections 3 presents our optimization model for energy efficiency in large-scale data centers along with our problem formulation and scheduling algorithm. Section 4 reports some experimental results and finally Section 5 concludes the paper.

## 2 Related Work

Advances in hardware technologies [5], such as manufacturing of low-power processors, and energy efficient memory modules, has reduced energy consumption in data centers. However, energy consumption of data centers is largely affected by resource management and utilization policies. Incorrect scheduling of resources without considering their utilization and application profile can considerably increase energy consumption. Much of the research in this area has focused on development of efficient

software approaches such as scheduling of VMs and consolidation of applications on fewer servers in data centers. Cardoso et al. [7] have leveraged min-max and shared parameters of virtualization software for placement and consolidation of VMs in a distributed virtualized environment. Obtaining these parameters for applications requires full knowledge about the priorities of applications, making their approach inapplicable to large-scale data centers. Verma et al. [8] have presented an application placement management for minimizing power costs while meeting performance constraints in virtualized heterogeneous systems. Srikantaiah et al. [9] have proposed a heuristic model for energy-aware consolidation. However, their algorithm does not guarantee to find a solution near the optimum state. Kusic et al. [10] have presented a framework for dynamic resource allocation using a look-ahead control method. Their approach addresses some of the challenges such as overhead due to the predictive configuration model. Petrousi et al. [11] have proposed a mixed integer-programming model (MIP) for power and performance optimization in virtualized servers.

### 3 The Proposed Approach

Increasing server utilization by consolidation of applications on fewer physical machines can be a significant opportunity for energy cost reduction in data centers. In this method, idle servers can be placed in offline mode. However, the concentration of resource demands on consolidated servers may increase execution time and degrade performance. Therefore, it is crucial to develop optimized methods to balance the performance and energy consumption in data centers.

To reach this tradeoff, we define an optimum point for processor utilization of each server and try to allocate applications to servers to minimize the number of active servers. At the same time, we try to minimize the sum of the distances of the current processor utilization in each server to the optimal point of that server. It should be noted that the optimum point is an upper threshold for servers' processor utilization, and that farther up that point may well degrade performance and reliability. Therefore, we set a penalty value for physical machines whose processor utilizations exceed the optimum point. Determining the optimum points in consolidation for energy efficiency, especially for large data centers such as Clouds with numerous servers, is challenging and requires a large number of experiments. In addition, the optimum consolidation point for servers varies depending on the type of workloads and performance degradation thresholds. The optimization process such as taguchi method [11] can be used to find the optimal point of resource utilization to address this issue. However, complete description of the optimization process is beyond the scope of this paper. In the next section, we present our model based on this approach.

### 4 The Optimization Model and Formulation

Our approach to a unified method for power and performance management in distributed environments is presented as a multi-objective optimization model.

The scheduling problem in general is proved to be NP-hard. Our solution to this problem exploits two sub-models, namely a power model and a migration model. In this section, we first describe these models and then formulate the problem.

#### 4.1 The Power Model

Since processor consumes a significant portion of a server's total energy [17], our power model focuses on consumed energy of processors. Considering a set of  $N$  number of servers and  $M$  number of VMs, each VM runs a particular application independently. Proposing an accurate power model for different allocation style of VMs on physical servers is difficult. We have conducted several experiments with different workloads and measured their power consumption using a Ziegler 3490 SS energy meter. According to obtained results, we estimated a linear relationship between processor utilization of servers and their power consumption. Let us assume that  $U_i$  is the processor utilization of a server that equals to the sum of processor utilization of VMs running on that server. Equation (1) formulates the power consumption  $P_i$  of a server at any given time.

$$P_i = (P_{\max} - P_{\min}) \times U_i + P_{\min} \quad (1)$$

where  $P_{\max}$  is the power consumption at the peak load (i.e. 100% processor utilization) and  $P_{\min}$  is the minimum consumed power in the active state. In addition, since the consumed energy to switch a physical machine to offline mode is a non-negligible amount of energy, we define a penalty value in our model to account for the cost of turning servers on and off.

#### 4.2 The Migration Model

In a consolidation plan, there is a need to migrate VMs from a source physical machine to a destination one. Since consumed energy due to VM migration is a significant amount, we do not ignore this value. Therefore, finding a model for migration cost of VMs and studying its effective parameters is essential. The cost of migration is independent of the background load and its processor utilization. It depends on the VM characteristics such as memory usage [8]. According to our experiments, for any two physical machines in our system and based on our virtualization infrastructure, i.e. KVM [13], Equation (2) calculates the consumed energy of live migration of VMs.

$$P_{\text{migration}} = 1.44 \times m + 0.34 \quad (2)$$

In Equation (2),  $P_{\text{migration}}$  denotes the consumed power or live migration of a VM and  $m$  denotes the VM's used memory size in Giga Bytes.

#### 4.3 The Problem Formulation

We can now formulate the problem using integer non-linear programming (INLP). In contrast to existing works, we have simultaneously considered both power cost and



performance in our optimization model. Our model includes two objective functions. Equations (3) and (4) formulate these objectives, respectively. The objective function given by Equation (3) relates to finding a resource allocation strategy to VMs that minimizes the overall cost in data centers in terms of energy consumption related to processor utilization, VM migration, and switching on and switching off servers. The second objective function given by Equation (4) relates to finding a mapping from VMs to servers that minimizes the sum of distance of the current processor utilization in each server to its optimal point. In order to prevent the processor utilization of servers to exceed from its optimum point, we insert a penalty value ( $F$ ) in this function. Table 1 shows the notations we have used in our optimization model.

**Table 1.** Notations used in the rest of paper

<b>Indices</b>	
$i$	Index of each server $i=1,2,\dots,M$
$j$	Index of each virtual machine $j=1,2,\dots,N$
<b>Variables</b>	
$X_{ij}$	Binary variable: 1 if virtual machine $j$ runs on server $i$ , otherwise 0
$Z_i$	Binary variable: 1 if server $i$ is on, otherwise 0
$F_i$	Binary variable: 1 if the total processor utilization on server $i$ is greater than $Optimum\_U_i$ , otherwise 0
<b>Parameters</b>	
$Cost\_onoff_i$	Cost of turning on and off the server $i$
$Cost\_Mig_j$	Cost of migration of virtual machine $j$ to any server
$Optimum\_U_i$	Optimal point of processor utilization in server $i$
$U_{ij}$	Processor utilization of virtual machine $j$ in the given server $i$
$Z_i$	Binary parameter: 1 if server $i$ is on, otherwise 0
$X_{ij}$	Binary parameter: 1 if virtual machine $j$ is running on server $i$ , otherwise 0
$a_i, b_i$	Power cost constants in server $i$
$L$	A very large number

Constraint (3) ensures that each VM has to be exactly allocated to one server. Constraint (4) prevents an undesirable solution. In this constraint, the sum of processor utilization of VMs running on a physical server is not allowed to exceed from the capacity of that server. It also guarantees that variable  $Z_i$  equals to one, if there is at least one VM running on server  $i$ . The problem solution is then given by the decision variables  $X_{ij}$ ,  $Z_i$  where  $X_{ij}$  determines the optimum allocation vector of VMs to servers

and  $Z_i$  denotes the state of servers (on or off). The two objective functions should be minimized simultaneously to attain maximum energy efficiency in a virtualized environment.

$$F_1: \quad Z_i' \times \left( \sum_{i=1}^M b_i + \sum_{i=1}^M a_i \times \sum_{j=1}^N U_{ij} \times X_{ij}' \right) + \sum_{i=1}^M Cost\_onoff_i \times |Z_i - Z_i'| + \sum_{i=1}^M Cost\_Mig_j \times |X_{ij} - X_{ij}'| / 2 \quad (3)$$

$$F_2: \quad \sum_{i=1}^M Z_i' \times (F_i \times L + (1 - F_i) \times (Optimum\_U_i - U_{ij})) \quad (4)$$

Subject to:

$$\sum_{i=1}^M X_{ij} = 1 \quad \forall j \in \{1, 2, \dots, N\} \quad (5)$$

$$\sum_{j=1}^N U_{ij} \times X_{ij} \leq Z_i' \quad \forall i \in \{1, 2, \dots, M\} \quad (6)$$

$$\sum_{j=1}^N (X_{ij} \times U_{ij}) > Optimum\_U_i - L \times F_i \quad \forall i \in \{1, 2, \dots, M\} \quad (7)$$

$$\sum_{j=1}^N (X_{ij} \times U_{ij}) \leq Optimum\_U_i + L \times (1 - F_i) \quad \forall i \in \{1, 2, \dots, M\} \quad (8)$$

We use the goal attainment method of Gembicki [14] to solve our presented multi-objective optimization problem. By applying this method, Equations (3) and (4) can be expressed as follows:

$$\begin{aligned} & \text{Min} \quad \varepsilon \\ & \text{Subject to :} \\ & F_1 - w_1 \times \varepsilon \leq F_1^* \\ & F_2 - w_2 \times \varepsilon \leq F_2^* \end{aligned}$$

where  $\varepsilon$  is an unrestricted scalar value,  $F_1^*$  and  $F_2^*$  are respectively the optimum solutions that are obtained from solving the objective functions  $F_1$  and  $F_2$  at the same time. The weighting vector  $[w_1, w_2]$  controls the relative degree of the objective functions. In this paper, the weighting vector  $w$  is set equal to the functions. The optimization model is a NP-hard problem that has remained unsolved to find solutions in large scale. To reduce the search space and time of planning, we use a simulated annealing heuristic to find the optimum solution of these functions. Figure 1 shows our optimization algorithm based on simulated annealing.

---

**Algorithm 1.** scheduling a set of VMs on a set of servers using the optimization model

---

**Input:** A current system state, indicating which VMs are running on which servers as  $X_{ij}$ , and their current processor utilization ( $U_j$ )

**Output:** The next system state, indicating which VMs should be executed on which servers

1. Start with a random VM allocation to servers, at known cost function value  $F_i^*$ ,  $T=temperature =hot, frozen=false$ ;
  2. While(!frozen)
    - 2.1. Repeat until cost function  $F_i$  stabilizes
      - 2.1.1. Perturb system slightly and compute the cost function value  $F_i$
      - 2.1.2. If ( $F_i < F_i^*$ ) Then accept new configuration unconditionally;  
 $F_i^*=F_i$ ;
      - 2.1.3. else if ( $rand() < e^{-\frac{(F_i^*-F_i)}{T}}$ ) Then accept new configuration
    - 2.2. If ( $(F_i - F_i^*)$  still decreasing over the last few temperatures) Then  $T=0.95T$ ; Else  $frozen=true$ ;
  3. return (final configuration as optimal solution)
- 

**Fig. 1.** The optimization algorithm based on simulated annealing heuristic

## 5 Evaluation

This section presents an evaluation of our proposed energy-aware scheduling algorithm based on metrics such as power reduction and dioxide carbon emission. In order to evaluate it on a large-scale virtualized system, a set of simulative experiments were set up as a method to evaluate the proposed algorithm. The described MATLAB implementation of our algorithm, described in Section 4, was used as a simulation test-bed. We collected a set of processor utilization traces from real VMs running real enterprise servers in a 120-minute period to benefit from the real characteristics of VMs. The traces were gathered from four different enterprise server types, namely database servers, web servers, file servers and finally enterprise mail servers. These four servers, exhibit different workload characteristics, so can be regarded as real world workloads. The model parameters such as  $Cost\_onoff_i$  in the  $F_1$  objective function presented by Equation (3) as well as power parameters were obtained experimentally. For any physical machines in our homogenous system and based on our virtualization infrastructure, i.e. KVM, the  $Cost\_onoff_i$ , and  $Optimum\_U_i$  parameter values in the  $F_1$  and  $F_2$  objective functions presented in Equations (3) and (4) were measured as 2.1Wh and 0.8, respectively. In addition, the values of power constant parameters ( $a$  and  $b$  used in Equation 3) in our system were measured as 0.2708 and 55.43, respectively. We deployed a set of VMs (ranging from 8 to 256) on our simulative

platform. On each data center configuration, we used three different VM assignment strategies. In the first approach, we used a static assignment of VMs to servers. In this method, the placement of VMs was performed by system administrator in priori and was not allowed to be altered during system execution. In the second approach, we used a consolidation method, wherein a central scheduler assigns VMs to servers in order to minimize the number of active servers and balance the load on servers. Finally, in the third method, we applied our proposed algorithm to schedule the VMs. To evaluate the efficiency of our algorithm, we measured three metrics, including power consumption, cost saving, and carbon dioxide emission.

Firstly, we measured the consumed power by three algorithms in each data center configuration. Figure 2 shows the results of our simulations. As this figure shows, for a nearly 120 minute execution, our proposed scheduler saves more energy than the other two methods. As an example, in a data center with 256 VMs, using our method consumes 34266.73 Watts, while the static approach consumes 42517 Watts. It means that in a 120-minutes time interval, our method nearly saves 8.25 KWh of energy, which is 24% saving on energy.

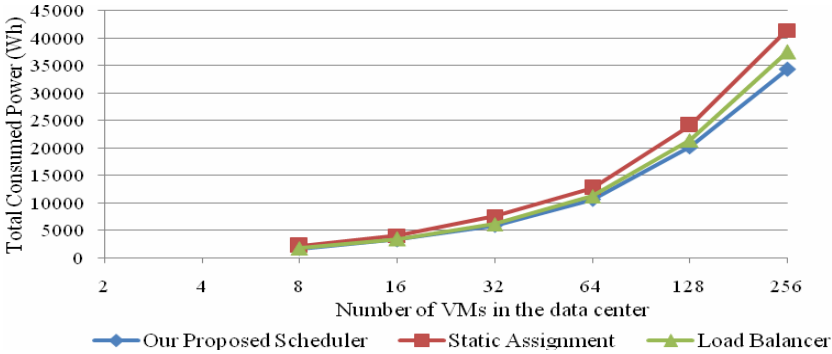


Fig. 2. The total consumed power on our simulative data center

To show the impact of our scheduling algorithm on carbon dioxide emission, we have used the average rate of carbon dioxide emission in the United States of America that was derived from the data published by the US Department of Energy [15]. Figure 3 shows the carbon dioxide emission amount by our proposed algorithm compared with two other algorithms in each data center configuration. As Fig. 3 indicates, in a data center with 256 VMs, the proposed algorithm can reduce the annual carbon dioxide emission to the environment by nearly 25 tons.

While our VM scheduling algorithm saves more energy, it leads to substantial amount of saving in electricity price. Figure 4 shows the cost savings obtained from our algorithm in different locations. To get these electricity cost savings, we varied the electricity price, while keeping all other factors such as the number of VMs the same as in a data center. The electricity prices were derived from the data published by the US Department of Energy and Energy Information Administration [16]. We modeled a typical data center with 1000 racks needing 10MW of power to operate

[16]. According to the electricity prices, for a country with an average electricity price, our algorithm saved almost 2M\$ per year. Clearly, this amount will be higher for countries with higher electricity prices.

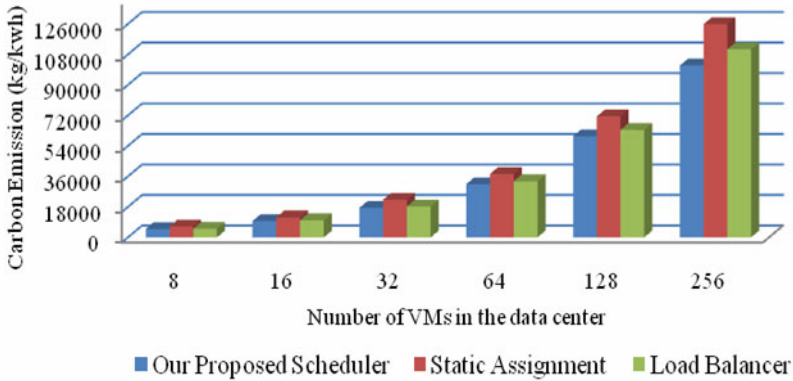


Fig. 3. The average carbon dioxide emission from our simulative data center

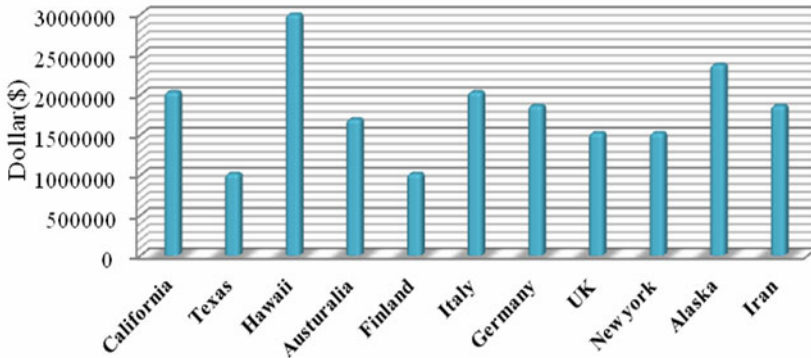


Fig. 4. The cost savings obtained from our algorithm in different locations

## 6 Conclusion

In this paper, we proposed a scheduling algorithm for dynamic placement of virtual machines on physical servers in a large-scale virtualized computing environment, considering both power cost and performance. To solve this optimization problem, we developed an integer non-linear programming formulation and showed how our approach can make a trade-off between energy cost and performance. Finally, the model was evaluated using experiments on a set of real traces of virtual machines. The simulative results of our model showed approximately 24% power savings compared to static methods. This value equals to an average of 2M\$ saving per year in a typical

data center. As a future work, we intend to incorporate other resources such as disk and network in the model and implement the mentioned algorithm on a virtualized data center.

## References

1. Horvath, T., Abdelzaher, T., Skadron, K., Liu, X.: Dynamic Voltage Scaling in Multi-Tier Web Servers with End-to-End Delay Control. *IEEE Transactions on Computers* (2007)
2. Uhlig, R., Neiger, G., Rodgers, D.: Intel Virtualization Technology. *Computer* 38(5), 48–56 (2005)
3. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a Consolidation Manager for Cluster. In: 5th International Conference on Virtual Execution Environments (2009)
4. Petrucci, V., Loques, O., Niteroi, B., Mosse, D.: Dynamic Configuration Support for Power-Aware Virtualized Server Clusters. In: 21th Euromicro Conference on Real-Time Systems, Dublin (2009)
5. Venkatachalam, V.: Franz. M.: Power Reduction Techniques for Microprocessor Systems. *ACM Computing Survey* (2005)
6. Lee, Y.C., Zomaya, A.Y.: Energy Efficient Utilization of Resources in Cloud Computing Systems. *Springer Journal of Supercomputing* (2010)
7. Cardoso, M., Korupolu, M., Singh, A.: Shares and Utilities based Power Consolidation in Virtualized Server Environments. In: Proceedings of IFIP/IEEE Integrated Network Management (IM 2009) (2009)
8. Verma, A., Ahuja, P., Neogi, A.: Pmapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. In: Issarny, V., Schantz, R. (eds.) *Middleware 2008*. LNCS, vol. 5346, pp. 243–264. Springer, Heidelberg (2008)
9. Srikantaiah, S., Kansal, A., Zhao, F.: Energy Aware Consolidation for Cloud Computing. In: *USENIX Workshop on Power Aware Computing and Systems* (2008)
10. Kusic, D., Kephart, J., Hanson, J., Kandasamy, N., Jiang, G.: Power and Performance Management of Virtualized Computing Environments via Lookahead Control. *Cluster Computing* (2009)
11. Petrucci, V., Loques, O., Mossé, D.: A Dynamic Configuration Model for Power-Efficient Virtualized Server Clusters. In: 11th Brazillian Workshop on Real-Time and Embedded Systems (WTR) (2009)
12. Taguchi, G.: *Introduction to Quality Engineering*. Mc Graw-Hill, New York (1990)
13. Kivity, A., Kamay, Y., Laor, D., Lublin, U., Liguori, A.: *Kvm: The Linux Virtual Machine Monitor* (2007)
14. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: *Optimization by Simulated Annealing*, Science. New Series (1983)
15. US Department of Energy, Voluntary reporting of greenhouse gases: Appendix F. Electricity emission factors (2007), [http://www.eia.doe.gov/oiaf/1605/pdf/Appendix20F\\_r071023.pdf](http://www.eia.doe.gov/oiaf/1605/pdf/Appendix20F_r071023.pdf)
16. US Department of Energy, US Energy Information Administration (EIA) report (2007), [http://www.eia.doe.gov/cneaf/electricity/epm/table5\\_6\\_a.html](http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html)
17. Rivoire, S., Shah, M.A., Ranganathan, P., Kozyrakis, C.: Joulesort: a Balanced Energy-Efficiency Benchmark. In: *SIGMOD, International Conference on Management of Data*. ACM, New York (200)

# A Leasing Instances Based Billing Model for Cloud Computing

Qin Yuan<sup>1</sup>, Zhixiang Liu<sup>1</sup>, Junjie Peng<sup>1,2,\*</sup>, Xing Wu<sup>1</sup>, Jiandun Li<sup>1</sup>, Fangfang Han<sup>1</sup>,  
Qing Li<sup>1</sup>, Wu Zhang<sup>1</sup>, Xinjin Fan<sup>3</sup>, and Shengyuan Kong<sup>3</sup>

<sup>1</sup> School of Computer Engineering & Science, Shanghai University,  
Yanchang Rd. 149, 200072 Shanghai, China

{yuanshiqin, lzxshu, jjie.peng, xingwu, jdli, fangfanghan,  
qli, wzhang}@shu.edu.cn

<sup>2</sup> Key Laboratory of Computer System and Architecture,  
Institute of Computing Technology, Chinese Academy of Sciences,  
Kexueyuan South Rd. 6, 100190 Beijing, China

<sup>3</sup> Zople Cloud Computing Technology Co., Ltd,  
Huma Rd. 571, 200431 Shanghai, China  
fan\_xinjin@hotmail.com, ksy28@bosideng.com

**Abstract.** As a new technology in IT industry, cloud computing has been much focused by both academia and industry. And many topics in cloud computing are under study. However, as one of the most important issue, billing and pricing has been not so much concerned. In this paper, we propose a novel pay-per-use billing model that can be used for the applications in cloud computing. Built based on leasing instances, the model is put forward according to the analysis of the real applications in the cloud computing environment. That is, with the model, the users will be charged totally by the amount of the resources they have used. Simulation results show that the model is correct and feasible.

**Keywords:** Billing model, Pricing, Leasing instances, Cloud computing.

## 1 Introduction

Cloud computing is an emerging innovation technology in IT industry. It enables all kinds of applications to obtain computing resources, storage resources, network resources and a variety of software services on the demand. As a new business model that may lead to many business opportunities, cloud computing technology has been drawing more and more attention. Amazon (EC2), Google (App Engine) and Microsoft (Windows Azure) have committed to the development of middleware and platforms of cloud computing. Many more others are focusing on the studies of virtualization, cloud storage, cloud security, load balancing, resource monitoring and so on, leaving billing and pricing alone with little focus.

---

\* Corresponding author.

In this paper, we try to present a novel pay-per-use for the cloud computing. The value of billing model lies in the economic benefit for cloud platforms and applications, and it is also an indispensable ingredient to almost every business model. Based on resource monitoring and use records, we can charge the applications in the cloud according to the resources consumption with a reasonable pricing method. Of course, to make benefit from this brand new business model, a reasonable pricing method is indispensable, this should and will be put on agenda as soon as possible.

## 2 Related Work

As for the billing mechanisms in the real applications, there are three common methods, i.e. year package, month package and hour package. Take Amazon EC2 cloud platform as an example, it mainly uses these three kinds of modes in the billing business model, which are called On-Demand Instances mode, Reserved Instances mode and Spot Instances mode[1].

- **On-Demand Instances:** On-Demand Instances enable user to purchase a pre-configured instance and pay by hours. It should be noticed is that each partial instance-hour will be billed as a full hour by the cloud provider.
- **Reserved Instances:** Reserved Instances leverage user to pay at one time for a pre-configured instance by year package, no matter this instance is at a running status or not. If user runs the instance, user would have to pay for the discounted usage charge based on hour package; otherwise, user will not need to pay usage charges for it. However, for windows instance, when an instance is running, the cost may be increased because of license. In this circumstance, user could adjust the discount rate charged per hour to achieve the same income, whereas the one-time costs are not affected. Amazon EC2 instances keep fees as Fig. 1 shows:

US – N. Virginia	US – N. California	EU – Ireland	APAC – Singapore	
<b>One-time Fee</b>				
<b>Standard Reserved Instances</b>	<b>1 yr Term</b>	<b>3 yr Term</b>	<b>Linux/UNIX Usage</b>	<b>Windows Usage</b>
Small (Default)	\$227.50	\$350	\$0.03 per hour	\$0.05 per hour
Large	\$910	\$1400	\$0.12 per hour	\$0.20 per hour
Extra Large	\$1820	\$2800	\$0.24 per hour	\$0.40 per hour
<b>Micro Reserved Instances</b>				
Micro	\$54	\$82	\$0.007 per hour	\$0.013 per hour
<b>High-Memory Reserved Instances</b>				

Fig. 1. Reserved instances

- **Spot Instances:** Spot Instances allow users to bid for unused Amazon EC2 capacity. The price fluctuates periodically depending on the relationship between supply and demand of the Spot Instance capacity.



From the example we can conclude that although Amazon EC2's model can avoid complex cloud accounting operations, it cannot guarantee fairness among the end users. Whereas billing according to resource usage could be more fair, however, nowadays there aren't appropriate business models and trust mechanisms, so it is difficult to apply in the real world. Thus, in this paper, we propose a novel business model and the corresponding the billing model [2].

### 3 Leasing Instances Model

Similar to Amazon we put forward our business model, leasing instances model. It is eligible for pay-per-use billing mechanism. To better put the model, we first introduce some related definitions and concepts, and then we give a running process and pre-conditions of the leasing instances model. Finally, we come up with the billing model, pay-per-use.

#### 3.1 Definition and Concept

**Primary User.** Primary user is the first customer who purchases Leasing Instances in the Cloud Computing Center (C3).

**Second User.** Second user purchases the Leasing Instances which primary user has already rent in the C3.

**Leasing Instances.** Leasing Instances are an extension of Amazon EC2 Reserved Instances. When they are idle, they will be submitted to C3 by primary user. The C3 can re-rent the instance to a second user. Primary user has to pay in full for the instance on year package and Pay by hour when running the instance with discount. The pricing method charging for primary user is the same as Reserved Instances do. The pricing method charging for second user is pay-per-use. Meanwhile, second user should allow the C3 to reclaim the instances at any time.

To primary user, there is generally long-term small and low urgency task. When primary user's Leasing Instance is rented, the primary user may be exempted from the additional cost on the corresponding time and even given cash compensation by the cloud provider.

To second user, there are many demands fit Leasing Instances, such as Anytime Algorithm [3], [4] that is a special iterative algorithm and can stop at any time. It includes Newton iteration, Wavelet Transform and so on. Provider's billing strategy is pay-per-use. Meanwhile, the charge is much lower than other business models do.

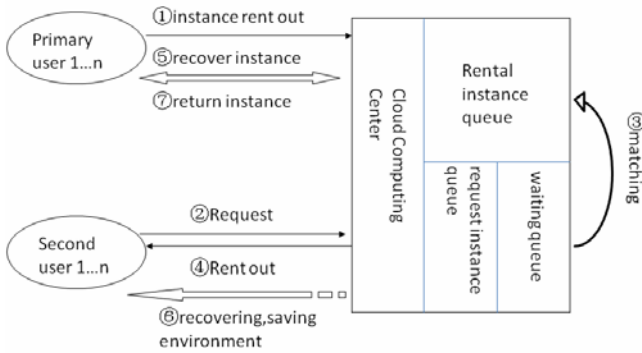
#### 3.2 Leasing Instance Process

Leasing instance process is as shown in fig. 2.

- **Step 1.** If primary user's instances are left unused, then the first primary user agrees to providers' lease agreement and submits the instance to C3. The C3 arranges instances on the basis of system type and configuration size in the different rental instance queue. For example, we may put a Win2008 instance in a queue and put a Redhat5. 4 instance on another queue. Then we will take

small-size instance to the front and big-size instance behind. Take Win2008 as an example, It assumes that there are two instances in the queue. Their configuration are: double core CPU (Frequency: 2.53GHz), 512MB RAM and double core CPU (Frequency: 2.53GHz), 1GB RAM. Now, we take the first instance to the front.

- **Step 2.** Second user submits instance applications according to his demands. The applications include system type (Win2008, Redhat5. 4, Ubuntu10. 10, etc.), instance configuration (number of CPU cores, memory size, disk size, Network bandwidth, etc.). The applications will be placed in the request instance queue.



**Fig. 2.** Leasing Instances process

- **Step 3.** The C3 uses front-to-back strategy to match the instances of the rental queue and the request queue. As long as instances of the rental queue achieve the instance requirements of the request queue (has got the same system type, the instance configuration in the rental queue bigger than or equal to the instance configuration in the request queue), this match is considered successful. If the request instance has not succeeded in matching instance of the rental queue, it will be put on the waiting queue. As for the instance in the waiting queue, when there are matches in the rental queue, the waiting instance will be assigned in priority to match. When the waiting time exceeds a threshold, C3 will assign a new instance to a second user.
- **Step 4.** If the leasing instance matches successfully, it will be assigned to a second user.
- **Step 5.** Primary user requests to recover the leasing instance that has been submitted to the C3.
- **Step 6.** If the instance has not been allocated, recover it directly. If the instance has been assigned to a second user, at this time, the C3 will need to preserve the running environment of the second user, and then recover the instance.
- **Step 7.** When step 6 finishes, the instance will be sent back to the primary user.

### 3.3 Preconditions

If you want to achieve the leasing instance model, the following two issues need to be addressed [5], [6].

**Security.** After primary user submitted an instance to the C3, we should ensure the safety of file on the instance. Our method is to guarantee that hard disk (not including the system disk) of the leasing instance can be safely mounted and unloaded. Additionally, we should also ensure that the data are also unchanged on disk. Therefore, in order to protect data security, when the primary user wants to lease the instance, we will unload the disk at first and then rent. To the system disk, we follow the Windows' strategy that takes second user as a guest user. And second user has no rights to access the system disk.

**Visibility.** Our billing model is pay-per-use to second user; it is not direct to users as month package does, so it might face users' doubt. Our strategy is to add a "cloud meter" on the client of second user. The cloud meter is similar to water meter or ammeter. Its function is to record and display the resources that second user has consumed. Meanwhile, we also add a simple cloud meter on the primary user client. Whose function is to record and display the time that the leasing instance has been used. So the "cloud meter" strategy will clarify the user's consumption and enhance the trust between the two parties.

## 4 Billing Model Based on Leasing Instances

The billing model discussed in this sector is proposed for second user using cloud computing resources which have been mentioned above. The model is proposed on the basis of the Anytime Algorithm (can cease at any time and the charging computing is iterating). Meanwhile, the instance price submitted by primary user can vary according to the number of second users. The demand relation can be simulated by an exponential function, without loss of generality, we use:

$$D(x_t) = \begin{cases} e^{\frac{x_t-1}{c}}, & x_t \in N, x_t \geq 2, \\ 1, & x_t = 0,1. \end{cases} \quad (1)$$

Where,  $C$  represents the maximum number of second users in the cloud resources pool,  $x_t$  is the number of second users in the cloud resources pool at the time, it varies with the time  $t$ . That is to say, the leasing instance price goes up with the increase of the number of competitive users.

As our model uses pre-paid policy, we have to evaluate the first computing price. We do statistics of the prices used in the similar problem and configuration (past prices stored in the database). We set the maximum one as the initial minimal computing price. That means the model won't compute unless the balance in user account is more than the initial minimal computing price. We use  $L$  to describe the initial

minimal computing price.  $k = \{1, 2, \dots, K\}$  is used to describe the available resources. Vector  $r = (r_1, r_2, \dots, r_k)$  is used to represent the quantity of resources. By means of the experiment analysis and observation the use of resources represents by a fluctuating curve as time goes by, so based on the dynamic billing principle, the price should be in proportion to the resource utilization ratio, and then the resource price is also represented by a fluctuating curve. To compute price for resource  $k$  in a certain time period more accurately, we use  $p_k = \int_{t_s^k}^{t_e^k} P_k(t) dt$ . And in practice, we

use  $p_k = \sum_{i=1}^n P_k \Delta t_i$ , where  $\Delta t_i = 1$  second. Then we define instance price function at the iteration of time  $t$  as formula (2) shows.

$$P_t = D(x_t) \cdot (1 + profit_{rate}) \cdot \sum_{k=1}^K r_k \cdot \int_{t_s^k}^{t_e^k} P_k(t) dt \cdot (t_e^k - t_s^k) \quad (2)$$

Where  $r_k$  is the quantity of resources second users apply for, such as the number of CPU (Frequency: 2.53GHz), size of memory, bandwidth, etc.  $t_e^k - t_s^k$  is the time resource  $k$  used at the iteration of time  $t$ .  $P_t$  is the instance price at the iteration of time  $t$ ,  $D(x_t)$  is the demand function,  $profit_{rate}$  is the resource provider rate.  $t_s^k$  is the start time of iteration of resource  $k$  at time  $t$  and  $t_e^k$  is the end time. Per-unit CPU, memory and bandwidth are defined above. In practice, the resource price can be defined as follows [7].

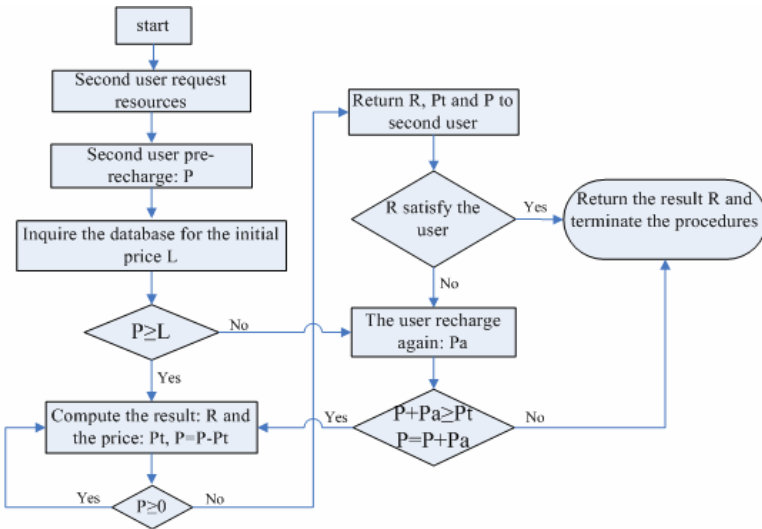


Fig. 3. Program flowing chart

$$P_k^c \geq V_{t_r}(X_{t_r}) - V_{t_r}(X_{t_r} - S_k^{t_p}) \approx \frac{\partial}{\partial x_k} V_t(X_{t_r}) =: \pi_{t_p}^k \tag{3}$$

The program based on Anytime Algorithm flowing chart is as fig. 3 shows. In the figure, the initial computing price  $L$  is evaluated from database (prices stored in it).

### 5 Experiment and Discussion

In order to prove the feasibility of billing model based on the leasing instance put forward in this paper, we simulate applications that fit Anytime Algorithm. Since there are too many Anytime Algorithm fit applications, without loss of generality, this paper uses the classic Newton iterative as an example for discussion. Newton iteration is a kind of computing following with the Anytime Algorithm, which means it can be stopped at anytime and the computing is iterated. In this experiment, Pre-recharge strategy is applied. That is the users will be charged beforehand. The UI is shown as fig. 4 demonstrates.

This simulation experiment uses the function  $f(x) = x^2 - 4 * x + 4$  as an example. Set the initial value 3, the recharging amount of the initial is 1000, when the recharging is not enough, users can recharge again. Meanwhile, assume the second user to apply 1 CPU (Frequency: 2.53GHz), 128MB RAM, 2MB bandwidth. If the C3 receives the recovery request of the primary user, it need to save the iteration result, and then recovers the instance from second user. After that, if another leasing instance meets the requirements of the second user, what is needed to do is to take the last result as the initial value of the new iteration. The final experiment data will be recorded in the database, the records are shown in Fig. 5.

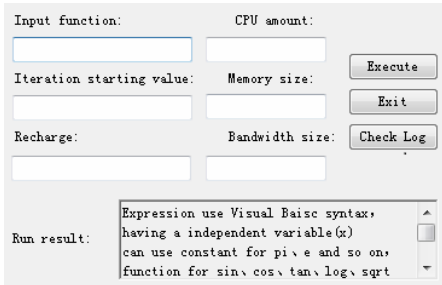


Fig. 4. Operation UI

id	count	result	price	timeGap
1	1	2.50019523623508	591.3668	4059
2	2	2.25029277820829	639.606	4405
3	3	2.1253413972604	626.538	4315
4	4	2.06286540433229	636.4116	4383
5	5	2.0316268085526	611.7276	4213
6	6	2.0160063080865	644.5420	4429
7	7	2.00819382651163	636.702	4385
8	8	2.00428333829856	606.7908	4179
9	9	2.00232065846893	624.2148	4299
10	10	2.00132750231421	611.8728	4214
11	0	2.00132750231421		0

Fig. 5. Experimental results

In figure 5, the col.2 shows iteration time. The col.3 shows the result of each iteration. The fourth shows the price, unit of the price is "unit price". The fifth one shows CPU time interval, its unit is microseconds.

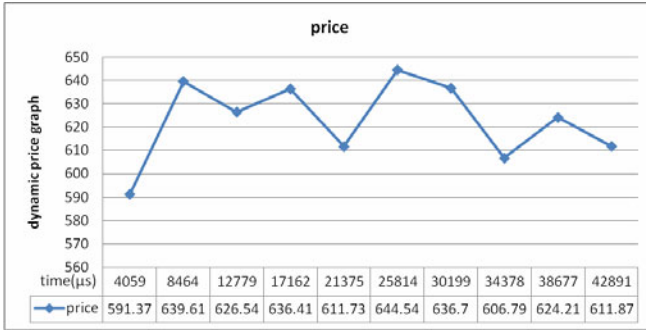


Fig. 6. Price fluctuation

In the process, in order to calculate conveniently and use easily, number of second users is simulated by a random integer from 1 to 100. The price fluctuation over time is shown in fig. 6. we assume that bandwidth price is a constant and use two sub-functions to simulate the CPU and memory price fluctuations. They are:

$$Cpu\ Price(t) = \begin{cases} 1.9, & t \in [4k, 4k + 10) \cup [4k + 20, 4k + 30) \\ 2.0, & t \in [4k + 10, 4k + 20) \\ 1.8, & t \in [4k + 30, 4k + 40) \end{cases} \quad (4)$$

$$Mem\ Price(t) = \begin{cases} 0.9, & t \in [4k, 4k + 10) \cup [4k + 20, 4k + 30) \\ 1.0, & t \in [4k + 10, 4k + 20) \\ 0.8, & t \in [4k + 30, 4k + 40) \end{cases} \quad (5)$$

## 6 Conclusion

The leasing instances model we have proposed in the paper provides a reference solution for cloud computing companies such as Amazon, Google, Microsoft and so on. It can help the cloud providers maximize the profits and expand the potential market without needing invest more for their resource infrastructure [7], [8]. The paper proposes a leasing instance based billing model and simulates the pricing method of the second user with the model. We also realize the pay-per-use method on leasing instances to solve the problem that price is changing with the fluctuation of the number of second users, efficiency of CPU and memory and bandwidth in the iterations. The simulation results show that the billing model based on leasing instances is feasible and has a real commercial value.

**Acknowledgments.** The authors are very thankful for the suggestions and comments from Prof. Yike Guo and the support from the Shanghai Leading Academic Discipline Project (No. J50103), the Key Laboratory of Computer System and Architecture

(Institute of Computing Technology, Chinese Academy of Sciences), the Ph.D. Programs Fund of Ministry of Education of China (No. 200802800007) and the Innovation fund of Shanghai University (No. B16010809007).

## References

1. Amazon web services website, <http://aws.amazon.com/ec2/pricing/>
2. Mihoob, A., Molina-Jimenez, C., Shrivastava, S.: A Case for Consumer-centric Resource Accounting Models. In: 3th IEEE International Conference on Cloud Computing, pp. 506–512 (2010)
3. Zilberstein, S.: Using Anytime Algorithms in Intelligent Systems. *AI Magazine* 17(3), 73–83 (1996)
4. Zilberstein, S.: Operational Rationality through Compilation of Anytime Algorithms. Ph.D. diss., Computer Science Division, University of California at Berkeley (1993)
5. Park, K.W., Park, S.K., Han, J., Park, K.H.: THEMIS: Towards Mutually Verifiable Billing Transactions in the Cloud Computing Environment. In: 3th IEEE International Conference on Cloud Computing, pp. 139–147 (2010)
6. Elmroth, E., Márquez, F.G., Henriksson, D., Ferrera, D.P.: Accounting and Billing for Federated Cloud Infrastructures. In: Eighth International Conference on Grid and Cooperative Computing, pp. 268–275 (2009)
7. Anandasivam, A., Premm, M.: Bid Price Control and Dynamic Pricing in Clouds (2010), <http://is2.lse.ac.uk/asp/aspecis/20090089.pdf>
8. Buyya, R., Abramson, D., Giddy, J.: Economic models for resource management and scheduling in Grid computing. In: *Concurrency and Computation: Practice and Experience*, pp. 1507–1542 (2002)

# A Scalable Multiprocessor Architecture for Pervasive Computing

Long Zheng<sup>1,2</sup>, Yanchao Lu<sup>3</sup>, Jingyu Zhou<sup>3,\*</sup>, Minyi Guo<sup>3</sup>, Hai Jin<sup>1</sup>,  
Song Guo<sup>2</sup>, Yao Shen<sup>3</sup>, Jiehan Zhou<sup>4</sup>, and Jukka Riekk<sup>4</sup>

<sup>1</sup> Huazhong University of Science and Technology, Wuhan, 430074, China

<sup>2</sup> The University of Aizu, Aizu-wakamatsu, 965-8580, Japan

<sup>3</sup> Shanghai Jiao Tong University, Shanghai, 200240, China

<sup>4</sup> University of Oulu, FIN-90014 Oulu, Finland

zhou-jy@cs.sjtu.edu.cn

**Abstract.** In a case study of a pervasive computing system, we have implemented a system for a JPEG encoding application. The previous system uses static deployment of computing resources, which limits computation capability. To address this limitation, this paper proposes a novel scalable architecture that allows extending a system by adding new subsystems, to meet increasing computation requirements. The novel architecture allows several subsystems to share their Resource Routers (RRs) and Processing Elements (PEs) to improve the efficiency of PEs by introducing a Share Degree (SD) mechanism. Experimental results show that when SD is equal to three, the system achieves the highest performance.

**Keywords:** ubiquitous multiprocessor, computing resource allocation, scalable.

## 1 Introduction

Olympus Future Creation Laboratory and University of Aizu have conducted a collaborative research on developing a general framework for the coming ubiquitous society, where a ubiquitous computing scenario named Ubiquitous Multi-Processor (UMP) that consists of many heterogeneous processing nodes has been extensively studied. A basic framework of multiprocessor simulation system has been implemented based on a multi-way cluster [7] and a double-buffered communication model [6] has been incorporated into the system that can improve the performance over 50%. M. Kubo et al. extends the system and implements a ubiquitous multi-processor network-based pipeline processing framework at the hardware simulation level to support the development of high performance pervasive applications [5].

Our previous work [2,11,3] did not consider the extensibility of the system, thus limiting computation capacity, because only one Resource Router (RR) manages all Processing Elements (PEs), which becomes a bottleneck during high

---

\* Corresponding author.



workload. This motivates us to design a new, scalable architecture to support increasing computation requirements. With the new architecture, computation capacity can be easily extended by simply adding new PEs and RRs. Specifically, our architecture consists of several subsystems. Each subsystem is composed the fixed number of PEs and a RR that manages these PEs. PEs are isolated by subsystems; and RRs are connected with the ring topology. This symmetric architecture allows us to add as many subsystems as needed to meet the computation requirement.

In order to increase the efficiency of PEs' usage, two mechanisms are proposed in our new architecture that are *Deploy as Needed* and *Share with Neighbors*. The latter mechanism allows several neighbor subsystems to share their RR and PEs. We introduce a Shared Degree (SD) to represents the number of neighbor subsystems that share their RRs and PEs with each other.

We model and analyze performance of the new architecture. Our analysis shows that SD affects both the efficiency of PEs' usage and the workload of RR, both of which eventually affect the performance of system positively and negatively, respectively. The optimal value of SD is the key to optimize the performance of our new architecture. With the performance model we construct, we can find the optimal SD theoretically. The similar thinking in performance model also appears in [8,4].

The remainder of this paper is structured as follows. Section 2 proposes our novel scalable multiprocessor architecture for pervasive computing system. Section 3 analyzes our novel architecture, introduces the SD and constructs a performance model. Performance evaluation is given in Section 4. Section 5 summarizes our findings and concludes this paper.

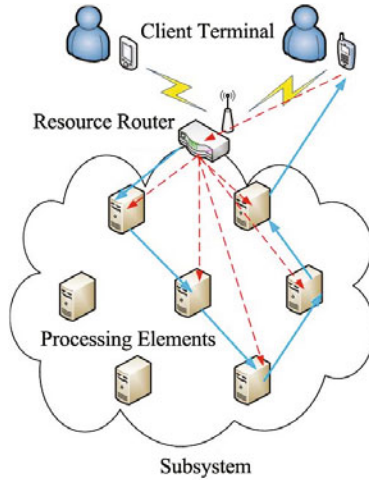
## 2 A Novel Scalable Multiprocessor Architecture

The UMP system has many PEs working as calculation nodes, and provides various functions by combining some PEs. This is very important to match a wide variety of users' needs and provide services they want.

In our earlier work, we have implemented a prototype UMP system that offers the JPEG encoding service for transforming from BMP into JPEG format. This paper takes this prototype as a case study. In the rest of this section, we first introduce the previous architecture of UMP, and then propose our new architecture.

### 2.1 The Drawbacks of Previous Architecture

In the previous architecture of UMP system, there are three kinds of nodes: Client Terminal (CT), PE, and RR, as illustrated in Fig. 1. A CT is usually a mobile device with which users can submit BMP pictures to a RR. The RR receives BMP pictures from different CTs and maintains these pictures in a task queue. RR processes each BMP picture in the task queue sequentially till the task queue is empty. Six PEs are managed by a RR to perform the transformation, since the transformation from BMP into JPEG has six phases and each PE is responsible for one phase.



**Fig. 1.** The previous architecture of UMP system

The previous architecture limits the scalability of a UMP system. Because there is only one RR managing all PEs that must request RR to transfer tasks to the next PEs, RR becomes the bottleneck of the whole UMP system during high workload and limits the number of PEs that can be deployed. Furthermore, because RR receives tasks from users, the solo RR design limits the physical area that the service of UMP system covers. The above drawbacks motivate us to design a new architecture to make the UMP system extensible.

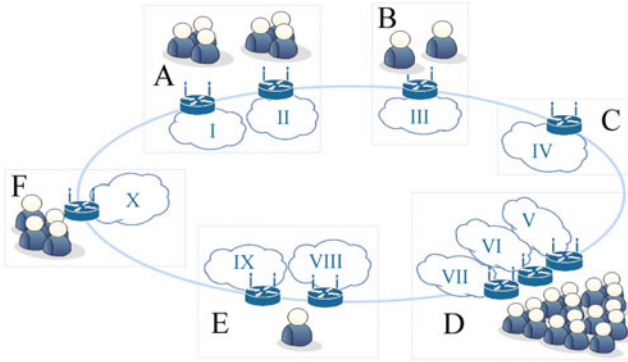
## 2.2 Overview of Our Architecture

We propose a novel architecture that is scalable for both computation capacity and service coverage. Fig. 2 illustrates the new scalable multiprocessor architecture for ubiquitous computing.

In this architecture, we have several solo-router UMP systems to meet computation capacity requirements. The solo-router UMP system in the new architecture is called a UMP *subsystem*: PEs in each UMP subsystem connect to their own RR; and RRs are connected in a ring topology. According to the demand, the subsystems are located in different places to increase service coverage. In Fig. 2, there are 10 subsystems distributed at location *A*, *B*, till *F*. Because the new architecture is decentralized and all subsystems are symmetric to each other, the new subsystems can be easily added when they are needed.

In our new architecture, the *Deploy as Needed* and *Share with Neighbors* mechanisms are used to guarantee the service quality for scalable usage and optimize the efficiency of the whole system.

**Deploy as Needed.** The *Deploy as Needed* mechanism is quite simple, which deploys subsystems at different locations based on daily statistics. For example, in a Theme Park, popular districts with more user demand will have more



**Fig. 2.** The scalable multiprocessor architecture

subsystems deployed. Less popular places may only require one subsystem to satisfy the requirement of service coverage. Thus, the *Deploy as Needed* is a static mechanism to satisfy user demand.

The real situation is more complicated. Take the Theme Park as example again; visitors are crowded in the district that has amusement rides in the day-time, but in the performance hall in the nighttime. Furthermore, when the animal march performance ends and 4D Cinema is about to start a stunning movie, lots of visitor move from the performance ground to the 4D cinema. Although we can put the appropriate amount of subsystems according to the statistical information, we also need a dynamic mechanism to handle peak loads.

**Share with Neighbors.** We focus on the *Share with Neighbors* mechanism that can deal with the peak load and use idle or under-loaded subsystems effectively. Specifically, we organize subsystems into a ring topology. The *Share Degree* represents the number of neighbor subsystems that an RR can use, including its own subsystem.

Take Fig. 2 as an example, where the ring direction is clockwise. When SD is set to three, the neighbors of Subsystem IV are subsystem IV, V, VI. When SD is one, subsystems cannot share their PEs with any other subsystems. When SD is equal to the number of subsystems in the ring topology, all subsystems can share their PEs.

When the *Share with Neighbors* mechanism is enabled, the PEs in the neighbors become a larger virtual subsystem, with RRs in the neighbors synchronized to each other. We propose Scalable Dynamic Allocation (SDA) algorithm to support the *Share with Neighbors* mechanism (See Algorithm 1). Note that the notation  $RRs$  specifies the RRs in its neighbor subsystems and PE1 means a PE in task phase 1. PE2, PE3, till PE6 refer to PEs in the subsequent phases. A task has six phases.

---

**Algorithm 1.** Scalable Dynamic Allocation (SDA) Algorithm
 

---

- 1: RR receives a new task from a CT.
  - 2: RR finds an idle PE1 from its subsystem and then transfers the task to this PE1.
  - 3: After getting the task RR, this PE1 sends a busy status message to *RRs*.
  - 4: After processing the task, this PE1 sends an idle status message to *RRs* and asks *RRs* for the next PE.
  - 5: *RRs* finds idle PE2 in each subsystem, and then tells the PE1.
  - 6: PE1 accepts the message of the nearest RR, ignores others. If there is no response from *RRs*, which means there is no available PE at this moment, PE1 waits a particular time, then asks again.
  - 7: PE2 sends the status busy to *RRs*; PE1 transfers the task to PE2, and sends idle status message to *RRs*.
  - 8: PE2, PE3, PE4, and PE5 act in the same manner.
  - 9: After PE6 has processed the task, PE6 transfers the processed task back to the RR that originally owns this task.
  - 10: RR sends back the processed task to the CT.
- 

### 3 Performance Model and Analysis

In our system, the tasks sent by users should be processed as soon as possible for good user experience. Therefore, our goal is to shorten the system response time, i.e., the time between sending a task to RR and getting the response. The average response time  $\mathcal{T}$  of all tasks in  $s$  phases during time interval  $\Delta$  can be expressed as

$$\mathcal{T} = \sum_{i=1}^s \frac{g(\Delta, i)}{\sum_{j=1}^s T^{i,j}} \sum_{j=1}^s T^{i,j} / \sum_{i=1}^s g(\Delta, i), \quad (1)$$

where  $T^{i,j}$  is the response time of the  $j$ -th task in the  $i$ -th phase, and  $g(\Delta, i)$  is the number of tasks of the  $i$ -th phase during the time interval  $\Delta$ .

Our performance model can be reasonably simplified with the following two assumptions on time interval  $\Delta$ :

1. The time interval  $\Delta$  is any interval after system initialization and before system halt;
2. Except for the beginning and end time, our system time is composed of many  $\Delta$  in which each subsystem has for each phase no more than one task requesting a PE for the next phase.

With the above assumption, it is easy to know that  $g(\Delta, i)$  is equal to the number of subsystems  $n$ . Furthermore, since each subsystem has the same number of phases  $s$ , so Equation (1) also can be simplified as:

$$\mathcal{T} = \sum_{i=1}^s \sum_{j=1}^n T^{i,j} / (n \cdot s) \quad (2)$$

In order to get  $\mathcal{T}$ , we first analyze  $T^{i,j}$ , which can be divided into:

$$T^{i,j} = T_{EXE}^{i,j} + T_{MSG}^{i,j} + T_{DATA}^{i,j} + T_{RT}^{i,j}, \quad (3)$$

where  $T_{EXE}^{i,j}$  is the time PE needs to execute the task in  $i$ -th phase and in  $j$ -th subsystem;  $T_{MSG}^{i,j}$  is the time that the PE needs to send messages to RRs and other PEs;  $T_{DATA}^{i,j}$  is the time needed to transmit the task in  $i$ -th phase and in  $j$ -th subsystem among users, RRs and PEs; and  $T_{RT}^{i,j}$  is the total delay time RRs need to response requests from PEs that executes the task in  $i$ -th phase and in  $j$ -th subsystem. Since  $T_{EXE}^{i,j}$  and  $T_{DATA}^{i,j}$  is determined by the size of tasks, we can consider them as a constant values  $C$ . Hence, (3) can be rewritten to

$$T^{i,j} = T_{MSG}^{i,j} + T_{RT}^{i,j} + C \quad (4)$$

So Equation (2) can be transformed into

$$\mathcal{T} = c + \left( \sum_{i=1}^s \sum_{j=1}^n T_{MSG}^{i,j} / (n \cdot s) \right) + \left( \sum_{i=1}^s \sum_{j=1}^n T_{RT}^{i,j} / (n \cdot s) \right), \quad (5)$$

where  $c$  is a constant, equal to  $C/(n \cdot s)$ . Let the average time of sending messages be  $\bar{T}_{MSG}$  and the average time of RRs response time be  $\bar{T}_{RT}$ , i.e.,  $\bar{T}_{MSG} = \sum_{i=1}^s \sum_{j=1}^n T_{MSG}^{i,j} / (n \cdot s)$ , and  $\bar{T}_{RT} = \sum_{i=1}^s \sum_{j=1}^n T_{RT}^{i,j} / (n \cdot s)$ .

Thus, the system performance is decided by  $\bar{T}_{MSG}$  and  $\bar{T}_{RT}$ . When SD increases, PEs have higher probability to find a free PE in the next phase, which leads to improved system performance. However, as SD increases, RRs will receive more requests from PEs and the response time of RRs will be longer, which leads to decreased system performance. In other words, the system performance is a function of SD and is a tradeoff between  $\bar{T}_{MSG}$  and  $\bar{T}_{RT}$ . Specifically,  $T_{MSG}^{i,j}$  can be calculated as:

$$T_{MSG}^{i,j}(d) = (P^{i,j}(d) \cdot \rho + (1 - P^{i,j}(d))) \cdot \theta(i), \quad (6)$$

where  $T_{MSG}^{i,j}(d)$  is the value of  $T_{MSG}^{i,j}$  when SD is  $d$ ,  $P^{i,j}(d)$  is the probability that a PE that executes a task in  $i$ -th phase  $j$ -th subsystem cannot find any free PE in next phase,  $\theta(i)$  is the time that a PE in the  $i$ -th phase need to find a free PE in the next phase by sending messages when there is at least one free PE in the next phase, and  $\rho$  indicates the average times a PE can find a free PE in the next phase after it fails to find a free PE in the next phase.

$P^{i,j}(d)$  can be calculated as

$$P^{i,j}(d) = \prod_{U^\alpha \in N(i,j,d)} \frac{S(U^\alpha)}{S_{max}}, \quad (7)$$

where  $U^\alpha$  represents a particular PE, function  $N(i,j,d)$  returns a set that includes all neighbor PEs of the  $j$ -th subsystem in the  $i$ -th phase when SD is  $d$ ,

$S(U^\alpha)$  is the task size of  $U^\alpha$ , and  $S_{max}$  is the maximal size of tasks. With Equation (7), we can get the probability that when SD is  $d$ , PE executing the task in  $i$ -th phase  $j$ -th subsystem cannot find any free PEs in the next phase in the worst case.

Now we get the expression of  $\bar{T}_{MSG}$ . Based on the analysis above, the average of RRs response time  $\bar{T}_{RT}$  is relative to the workload of each RR, so we first calculate  $W_{RT}^{i,j}(d)$ , the workload of RR of the  $j$ -th subsystem in the  $i$ -th phase when SD is  $d$ .

$$W_{RT}^{i,j}(d) = (P^{i,j}(d) \cdot \rho + (1 - P^{i,j}(d))) \cdot \omega(i), \quad (8)$$

where  $\omega(i)$  is the workload that the PE executing the task in  $i$ -th phase  $j$ -th subsystem causes to find the free PE in the next phase if there is at least one free PE in the next phase.

In our architecture, there are  $n$  subsystems in the system, that is,  $n$  RRs exist. The workload of RR in the  $j$ -th subsystem,  $R^j$  can be calculated as follows:

$$R^j = \sum_{k=j}^{k+d-1} \sum_{i=1}^s W_{RT}^{i,k}(d). \quad (9)$$

Because  $k + d - 1$  may be greater than  $n$ , we define  $W_{RT}^{i,j} = W_{RT}^{i,j+n}$  for ( $j < n$ ).

Hence, if  $\mu$ , the processing capacity of RR during  $\Delta$  is known, the average of response time of RRs,  $\bar{T}_{RT}$  can be calculated as

$$\bar{T}_{RT} = \frac{1}{2 \cdot \mu \cdot n} \sum_{j=1}^n R^j \quad (10)$$

Finally, we re-organize the expressions above. The average of response time of each task  $\mathcal{T}$  is expressed as follows.

$$\left\{ \begin{array}{l} \mathcal{T} = \bar{T}_{MSG} + \bar{T}_{RT} + c \\ \bar{T}_{MSG} = \frac{1}{n \cdot s} \sum_{i=1}^s \sum_{j=1}^n T_{MSG}^{i,j}(d) \\ \bar{T}_{RT} = \frac{1}{2 \cdot \mu \cdot n} \sum_{j=1}^n \sum_{k=j}^{k+d-1} \sum_{i=1}^s W_{RT}^{i,k}(d) \\ T_{MSG}^{i,j}(d) = (P^{i,j}(d) \cdot \rho + (1 - P^{i,j}(d))) \cdot \theta(i) \\ W_{RT}^{i,j}(d) = (P^{i,j}(d) \cdot \rho + (1 - P^{i,j}(d))) \cdot \omega(i) \\ P^{i,j}(d) = \prod_{U^\alpha \in N(i,j,d)} \frac{S(U^\alpha)}{S_{max}} \end{array} \right. \quad (11)$$

In Equation (11), the variables  $s$ ,  $n$ ,  $\rho$ ,  $\mu$ ,  $\theta(i)$  and  $\omega(i)$  are known. In these variables,  $s$ ,  $n$ ,  $\rho$ , and  $\mu$  are system parameters;  $\theta(i)$  and  $\omega(i)$  are decided by the SDA algorithm in the system. From the analysis of SDA algorithm, it is easy to

**Table 1.**  $\theta(i)$  and  $i$ 

$i$	1	2	3	4	5	6
$\theta(i)$	$2\tau$	$3\tau$	$3\tau$	$3\tau$	$3\tau$	$2\tau$

**Table 2.**  $\omega(i)$  and  $i$ 

$i$	1	2	3	4	5	6
$\omega(i)$	3	2	2	2	2	0

define a map between  $\theta(i)$  and  $i$ ; so is the one between  $\omega(i)$  and  $i$ . The maps are listed in Table 1 and Table 2.

Therefore, we find out that  $\mathcal{T}$  is decided by  $d$ , the value of SD. With the performance model expressed by Equation (11), we can find the optimal value of SD, such that the system response time is the minimal.

**Table 3.** Parameters of Practical Environment

Symbols	Values	Descriptions
$s$	6	The number of phases.
$n$	10	The number of subsystems.
$\mu$	$2.66 \times 10^5$ req/s	The process capacity of RR. (Gigabit Router using 266MHz embedded processor.)
$\tau$	0.5 ms	The message transmission delay.
$d$	$[1, n]$	The range of values of SD.
$S(U^\alpha)/S_{max}$	$[0.8, 1]$	The range of $S(U^\alpha)/S_{max}$ .
$\Delta$	0.5 ms	The value that satisfies the two assumptions about the time interval in practice.
	60	The number of PEs in a subsystem. This value will be used indirectly in Equation (7).

## 4 Performance Evaluation

We evaluate the performance of our systems with simulations. The parameters for the system are listed in Table 3. Our simulation is implemented using Matlab.

Fig. 3 shows the values of  $\bar{T}_{MSG}$  and  $\bar{T}_{RT}$  with different values of SD. When  $\rho$  increases, both  $\bar{T}_{MSG}$  and  $\bar{T}_{RT}$  become larger. However, the difference between values of  $\rho$  is very small, and the difference is small when SD is greater than four. When SD varies from 1 to 10,  $\bar{T}_{MSG}$  decreases but  $\bar{T}_{RT}$  increases, which is consistent with our analysis in Section 3.

Fig. 4 depicts the values of  $\mathcal{T}$  with different values of SD. Since there is a constant  $c$  in  $\mathcal{T}$ , we show the value of  $\mathcal{T}$  excluding  $c$ . We can observe that the average response time decreases first, and then increases as the value of SD varies from 1 to 10. For different values of  $\rho$ , all the average response times achieve the minimum when SD is three. Therefore, with our current system configuration, we should set SD to three to minimize the average response time.

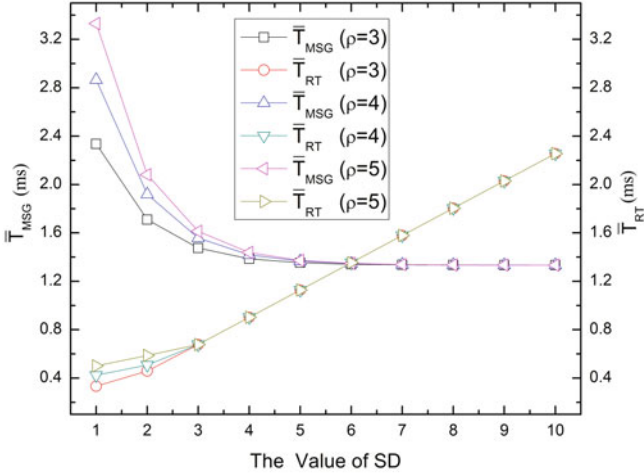


Fig. 3. The values of  $\bar{T}_{MSG}$  and  $\bar{T}_{RT}$  with different values of SD

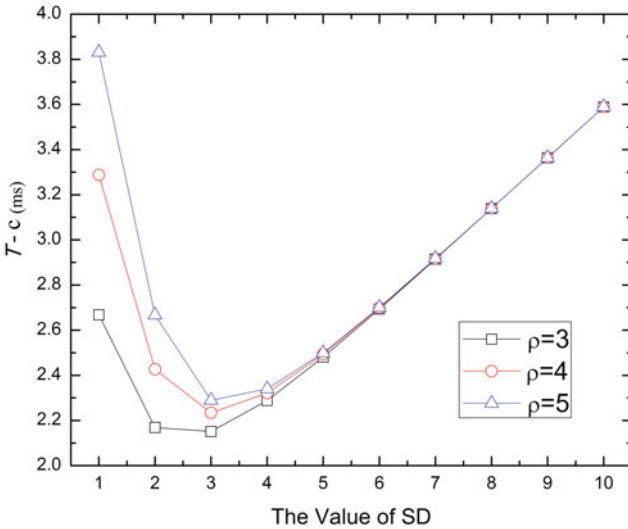


Fig. 4. The values of  $T$  with different values of SD

## 5 Conclusion

Our previous work uses the static deployment of computing resources, which is difficult to extend for use in dynamic environment. To solve this problem, we propose a novel scalable architecture with a ring topology to connect each subsystem so that this decentralized structure makes it easier to extend the computing capacity by adding the new subsystem to any location of the system.



Our architecture also enables *Deploy as Needed* and *Share with Neighbors* — two mechanisms that optimize the performance. We focus on the *Share with Neighbors* mechanism and propose a new SDA algorithm. To achieve the best performance, we formally analyze our system performance and find that the performance of our system depends on the value of SD. Experimental results show that with our current system configuration, when SD is set to three, the average response time is lowest.

## Acknowledgment

This work was supported in part by NFSC (Grant No. 60811130528, 61003012), and the National 973 Basic Research Program (No. 2007CB310900) of China. Jingyu Zhou is the corresponding author.

## References

1. Dong, M., Zheng, L., Ota, K., Guo, S., Guo, M., Li, L.: Improved Resource Allocation Algorithms for Practical Image Encoding in a Ubiquitous Computing Environment. *Journal of Computers* 4(9), 873–880 (2009)
2. Dong, M., Zheng, L., Ota, K., Guo, S., Guo, M., Li, L.: A trade-off approach to optimal resource allocation algorithm with cache technology in ubiquitous computing environment. In: *International Conference on Computational Science and Engineering, CSE 2009*, vol. 1, pp. 9–15 (August 2009)
3. Dong, M., Zheng, L., Ota, K., Ma, J., Guo, S., Guo, M.: A probabilistic-approach based resource allocation algorithm in pervasive computing systems. In: *International Conference on Computer Application and System Modeling (ICCASM)*, vol. 11, pp. V-315–V-319 (October 2010)
4. Huh, J., Kim, C., Shafi, H., Zhang, L., Burger, D., Keckler, S.W.: A nuca substrate for flexible cmp cache sharing. *IEEE Transactions on Parallel and Distributed Systems* 18, 1028–1040 (2007)
5. Kubo, M., Ye, B., Shinozaki, A., Guo, M.: Ump-percomp: A ubiquitous multiprocessor network-based pipeline processing framework for pervasive computing environments. In: *21st International Conference on Advanced Information Networking and Applications, AINA 2007*, pp. 611–618 (2007)
6. Shinozaki, A., Shima, M., Guo, M., Kubo, M.: Multiprocessor Simulator System Based on Multi-way Cluster Using Double-buffered Model. In: *21st International Conference on Advanced Information Networking and Applications, AINA 2007*, pp. 893–900 (2007)
7. Shinozaki, A., Shima, M., Guo, M., Kubo, M.: A high performance simulator system for a multiprocessor system based on a multi-way cluster. In: Jesshope, C., Egan, C. (eds.) *ACSAC 2006*. LNCS, vol. 4186, pp. 231–243. Springer, Heidelberg (2006)
8. Zheng, L., Dong, M., Jin, H., Guo, M., Guo, S., Tu, X.: The core degree based tag reduction on chip multiprocessor to balance energy saving and performance overhead. In: Ding, C., Shao, Z., Zheng, R. (eds.) *NPC 2010*. LNCS, vol. 6289, pp. 358–372. Springer, Heidelberg (2010)

# Enhancing the Reliability of SIP Service in Large-Scale P2P-SIP Networks

Fei Xu, Hai Jin, Xiaofei Liao, and Fei Qiu

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.** Recently P2P-SIP (*Peer-to-Peer Session Initiation Protocol*) has been proposed to improve the scalability and reliability of the traditional SIP (*Session Initiation Protocol*) networks. However, P2P-SIP makes SIP service unreliable because large-scale P2P-SIP networks are probably more dynamic than traditional SIP networks and the service nodes are very likely to fail or leave the P2P-SIP networks. To deal with this issue, we propose a novel and lightweight algorithm, which replicates SIP transactions information among the nodes in P2P-SIP networks and selects one of the successors of the failed or departed node as the takeover server. Moreover, to reduce the retrieving delay of SIP transactions replicas, we optimize our algorithm by storing these replicas directly in the successors of the failed or departed node. The simulation results demonstrate that our algorithm maintains 99% dialogs correctly in the presence of nodes failure with acceptable overheads over the Internet.

**Keywords:** SIP, P2P-SIP, Reliability, SIP service.

## 1 Introduction

Recently P2P-SIP has been proposed to improve the scalability and reliability of the SIP telephony networks. Some implementations of P2P-SIP, such as SIPPEER [10], P2PNS [1], and RELOAD [3], have achieved better performance than traditional SIP networks. P2P-SIP is being designed by the P2PSIP working group of the Internet Engineering Task Force [15]. However, large-scale P2P-SIP networks are probably more dynamic than traditional SIP networks and the service nodes are very likely to fail or leave the P2P-SIP networks when they are offering SIP service. The departure or breakdown of the service node has side effect on P2P-SIP networks, corrupting the reliability of SIP service.

The unreliability of SIP service is a common issue in SIP networks, and it is mainly due to the breakdown or departure of the SIP serving node in SIP networks. For example, after a session established between the caller and the callee, the failure or departure of caller's registrar results in several harmful consequences. First, the downstream SIP elements cannot change their working states and release the memory

of the corresponding dialog. Second, the users' transactions information is lost, and the service provider cannot control the dialog and exactly charge the caller for this dialog in a right way. Last but not the least, the subsequent SIP signaling (if there is any) cannot reach its correct destination, which may annoy the users and impact the users' experience of SIP service. One may declare that this problem could simply be fixed by the simple timeout mechanism, and we argue that the timeout mechanism only addresses the first problem, while the last two (the lost of users' transactions information and the signaling transmission problem) problems cannot be solved just by the simple timeout mechanism. So, what we need is maintaining the reliability of the SIP service during the call at a reasonable cost.

The traditional SIP telephony networks address the problem related to SIP service reliability by using the traditional redundancy and failover methods, such as reliable server pooling [4] and IP address takeover [12]. However, these solutions may be not suitable for large-scale P2P-SIP networks because the cost of the two techniques is too high for service providers, which will be discussed in Section 5 in detail. Therefore, in P2P-SIP scenarios, we have to explore a better method to handle the unreliable SIP service problem caused by the nodes failure or departure. One naive approach for maintaining the reliability of SIP service would be to replicate the SIP routing path by sending one SIP message to two SIP servers. We contend that such an approach is not acceptable for large-scale P2P environments because of the associated high performance overhead.

In this paper, we consider that if we leverage the underlying P2P systems to provide the reliability of SIP service for the upper SIP layer, it would be a lightweight method without adding extra nodes in P2P-SIP networks. This motivates the creation of our algorithm, which simply replicates SIP transactions information among the nodes in P2P-SIP networks and selects the appropriate takeover server when a SIP server failure occurs. In order to reduce the retrieving delay of SIP transactions replicas, we optimize our algorithm by storing SIP transactions replicas in the successors of the failed or departed node. Even in call busy hours, our algorithm is able to maintain 99% dialogs correctly in the presence of nodes failure or departure.

In section 2 we discuss our motivation, assumptions and requirements of this paper. Our detailed algorithm (SIP transactions replication and runtime failover) is introduced in section 3. We illustrate the severity of unreliable SIP service provided by original P2P-SIP, simulate our algorithm and evaluate its performance in section 4. We introduce the related work and compare the previous works with our solution in section 5 and we conclude the paper and show our future work in section 6.

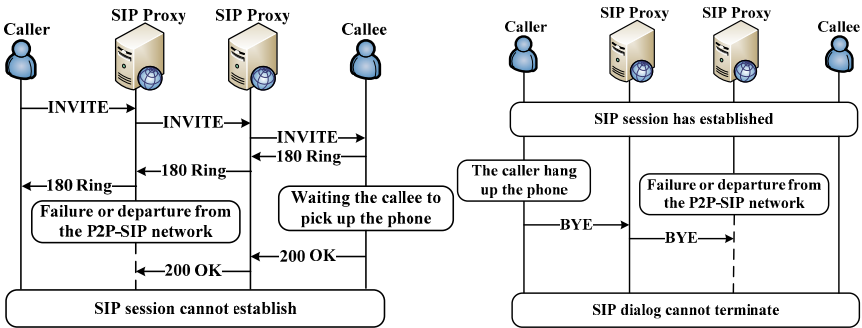
## 2 Background

In this paper, we adopt SIP over P2P [14] model for our P2P-SIP implementation. In this model, each node implements two modules: SIP module and DHT (*Distributed Hash Table*) module. The SIP module provides users SIP service including SIP registration and SIP call setup, whereas the DHT module provides location and storage service for SIP module by means of P2P overlay.

### 2.1 Motivation

In call leisure hours around 5% dialogs fail in the presence of 30% nodes failure or departure in our simulation, which will be discussed in section 4.2; while in call busy hours our simulation reveals that around 20% dialogs fail when 30% nodes fail or depart from the networks, which severely impacts the users' experience.

Fig. 1 illustrates two scenarios for unreliability of SIP service. In Fig. 1a, the SIP 200 OK response fails to reach the caller since the caller's registrar fails or departs from the networks when the callee's phone is ringing, directly resulting in the failure of SIP session establishment. Fig. 1b shows another scenario in which the callee's registrar fails or departs from the networks, which causes the failure of SIP session termination. These two scenarios illustrate the reasons for the unreliability of SIP service in P2P-SIP networks.



**Fig. 1.** Two illustrative scenarios for unreliability of SIP service: the SIP node fails or departs from the networks a) when the callee's phone is ringing, b) after SIP session has established

According to the two scenarios we present above, we can conclude that the existing P2P-SIP implementations have at least three defects when facing the failure or departure of the serving nodes:

1. The downstream SIP elements (including SIP proxies and agents) cannot release the memory of the corresponding SIP transactions and change their working states.
2. The SIP transactions information is lost, so the integrity of SIP transactions cannot be ensured.
3. The SIP signaling cannot be forwarded to the destination correctly, causing the unreliability of the SIP signaling transmission.

### 2.2 Assumptions and Requirements

We only focus on the problems caused by the departure or breakdown of SIP proxies in P2P-SIP networks, and we do not consider the DHT lookup failure caused by the dynamic P2P networks; it is another common issue, and has been well studied in [6]. We use Chord [11] as our DHT algorithm of P2P-SIP for a good illustration.

Reliability is one of the most important requirements because the unstable service has significant consequences and impacts the user's trust to the service provider, and we have to enhance the reliability of SIP service as best as we can.

The performance and the cost are the other two important requirements. We should maintain the reliability of the SIP service at a low cost and the performance of the SIP service should be preserved at an acceptable level.

### 3 Algorithm Design

Our algorithm is divided into two parts: replication of SIP transactions (section 3.1) and the runtime failover (section 3.2).

#### 3.1 Replication of SIP Transactions

In P2P systems, it is convenient to achieve the reliability of information by keeping several information replicas in P2P overlay, so we only need to consider the data structure of the stored SIP transactions entry.

The identifier of SIP transactions can be defined as the string combination of three elements (branch parameter in Via field, sent-by value in Via field, and the method of SIP message). Typically, SIP transactions contain two kinds of information: one is the state in which the transactions stay, and the other is SIP message packets which the transactions send finally [8]. SIP transactions can be divided into two kinds: server transactions and client transactions. In our approach, we only keep the replications of the client transactions in P2P overlay.

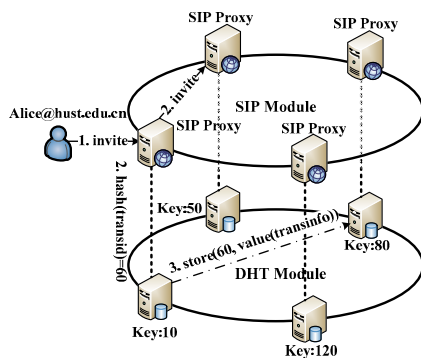


Fig. 2. Replication of INVITE transaction in P2P overlay

Recall that the information stored in P2P systems is in  $(key, value)$  format. We use the hash value of the aforementioned transaction identifier as the key, and distribute the transactions information (the string combination of SIP transactions state and the last sent SIP message) among the DHT nodes.

We give an example of replicating SIP INVITE transaction in Fig. 2. After receiving a SIP INVITE request, the SIP proxy creates the SIP transactions locally and

informs the DHT module to store the SIP transactions in the background. The DHT module then distributes the formatted (*key, value*) entry in P2P overlay. The storing procedure is asynchronous with the subsequent SIP procedure.

### 3.2 Runtime Failover

Runtime failover can be divided into three steps: the message sender 1) detects faults, 2) selects the takeover server. 3) The takeover server does failover routine.

**Fault detection mechanism.** We implement the fault detection mechanism on the SIP transaction layer and use UDP as the transport protocol. When the transaction layer sends out an unreliable SIP message<sup>1</sup>, it starts a temporary timer, and waits for a temporary acknowledgement response which is immediately sent back by the SIP message receiver. The temporary timer is killed immediately after the message sender receives the temporary acknowledgement response. If the temporary timer expires, the transaction layer considers the next hop of the SIP message as a failed or departed node and starts the routine for selecting a takeover server.

**Runtime selection for the takeover server.** After locating the exact failed node on the message routing path, the message sender first gets the IP address of the next hop (the failed or departed node) in the message, and calculates the hash value of the IP address of the next hop. Then the IP address of the takeover server can be obtained by finding the successor of the failed or departed node. Before sending out the message to the takeover server, we need to check whether the takeover server is the message sender or the next hop of the message and do some modifications to the message.

1. If false, we only need forward the message to the takeover server if the message is a SIP request. If the message is a SIP response, we have to alter the Via lists and Record-Route sets of the message in order to make the subsequent SIP messages be routed to the takeover server correctly.
2. If true, we need delete the first Via entry in Via lists and the corresponding Record-Route entry in Record-Route sets if the message is a SIP response. If the message is a SIP request, only the first Route entry in Route sets should be deleted.

After the corresponding modifications have been done to the message, the updated message is sent to the takeover server.

**Takeover server routine.** After receiving the updated SIP message, the takeover server should check the type of the SIP message. If the message is a SIP request, the server follows the ordinary routine to handle the message. If the message is a SIP response, the server has to retrieve the transactions information replicas in P2P overlay and load the transactions information into main memory. At last, the takeover server processes the updated message with the ordinary routine of the SIP server.

Fig. 3 illustrates an example for runtime failover to reliably transmit a 200 OK response when a fault occurs on the SIP response routing path. Step 1 to step 3 represent the fault detection mechanism. Step 4 to step 6 show the runtime selection for

---

<sup>1</sup> ACK, BYE requests and responses to INVITE, BYE requests are unreliable SIP messages.

takeover server. Step 7 to step 9 describe the takeover server routine after receiving the updated SIP message.

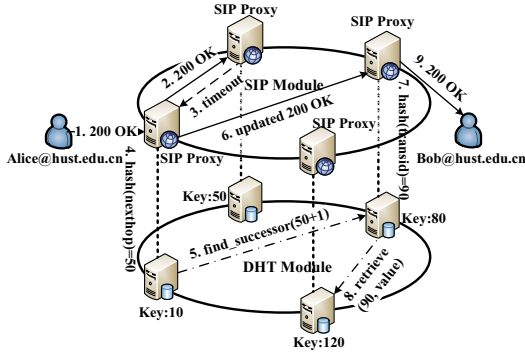


Fig. 3. Runtime failover to reliably transmit a 200 OK response

**Optimization.** After receiving the updated SIP message, the takeover server may still have to wait some time for retrieving the transactions information replicas from the other nodes in P2P overlay, resulting in an additional networks transmission delay. Therefore, we optimize our algorithm by storing the transactions replicas in the successors of the node because one of the successors may be chosen as the takeover server when a failure occurs. In this way, the takeover server could retrieve the transactions replicas directly from local disks, which reduces the replicas retrieving latency overhead.

## 4 Simulation and Evaluation

We simulate P2P-SIP protocol and our algorithm based on NS2 (*Network Simulator 2*) platform [13], and we choose NS 2.34 which is the update-to-date version of NS2. We implement the P2P-SIP protocol in NS 2.34 by reusing the SIP module in [7] and adding our algorithm to the original P2P-SIP, and we refer to it as the reliable P2P-SIP implementation.

### 4.1 Simulation Topology

By using the GT-ITM topology generator provided by NS2, we generate a simple simulation topology, which has 104 SIP proxies and 400 user agents within two transit domains and eight stub domains. Two transit domains represent two separate P2P-SIP networks, and each P2P-SIP network (transit domain) has four stub domains. We assign 200 users for each transit domain, and start 400 inter-domain and 600 intra-domain sessions.

We compare our reliable P2P-SIP implementation with the original P2P-SIP in section 4.2 and evaluate the failover overhead caused by our algorithm in section 4.3.

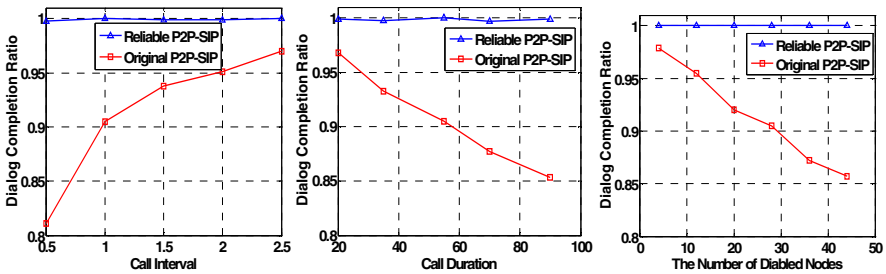
### 4.2 Dialog Completion Ratio

Dialog completion ratio is defined as the number of successfully completed dialogs divided by the total number of initiated dialogs in a certain period of time. This value can be used to indicate the quality of SIP service during a period of time.

In order to illustrate the severity of unreliable SIP service provided by the original P2P-SIP and the effectiveness of our algorithm, we run three sets of simulations for both the original P2P-SIP and our reliable P2P-SIP. In our simulation, we uniformly set the disabled nodes among the network and the node disabling time according to the overall simulation time. We have three parameters: call interval, call duration, and the number of disabled nodes. For each set of simulation, we run the overall 1000 sessions with one parameter varying and the other two statically configured to typical values. Table 1 lists our simulation parameters.

**Table 1.** Simulation parameters

	Call interval/seconds	Call duration/seconds	Disabled nodes
Typical value	1	55	28
Value range	0.5, 1, 1.5, 2, 2.5	20, 35, 55, 70, 90	4, 12, 20, 28, 36, 44



**Fig. 4.** Two results of simulations for a) five different call intervals, b) five different call durations, c) for six different numbers of disabled nodes

Fig. 4 indicates that the dialog completion ratio is related to the call interval, the call duration and the number of disabled nodes. In call busy hours, the call interval becomes smaller and the call duration gets longer, and the dialog completion ratio for the original P2P-SIP decreases rapidly shown in Fig. 4 a) and b). Fig. 4c shows that the dialog completion ratio for the original P2P-SIP decreases as the number of disabled node increases. The result may get worse in real P2P-SIP networks because the real calling situation is more complex and worse than that we set in our simulation (e.g. there are concurrent calls). On the other hand, the reliable P2P-SIP maintains around 99% dialogs correctly despite of the changes of the call interval, the call duration and the number of disabled nodes, which demonstrates that our algorithm ensures the reliability of SIP service in P2P-SIP networks.



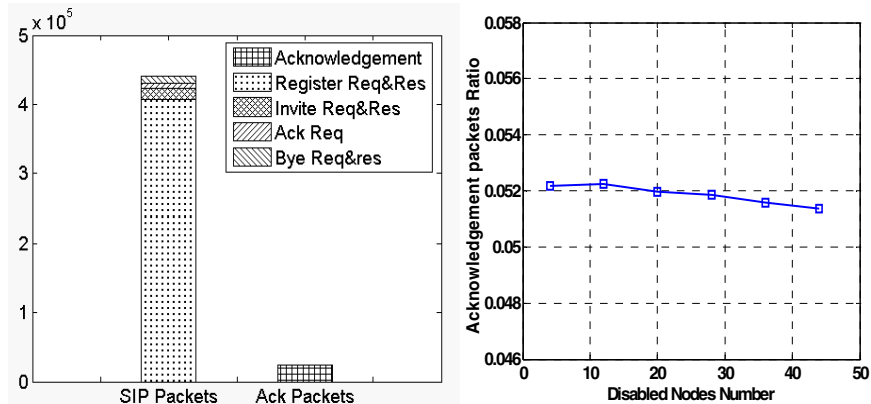
### 4.3 Failover Overhead

Our algorithm brings on several failover overheads. Here we mainly analyze and evaluate packets overhead.

**Packets overhead.** As described in section 3.2, our approach needs a temporary acknowledgement response from message receiver for each unreliable SIP message, resulting in additional packets overhead over the networks.

In our simulation, we trace all the message packets including the original SIP messages and the acknowledgement responses. In order to illustrate the acknowledgement packets overhead, we first run our simulation in the typical scenario (the call interval is set to 1 seconds; call duration is set to 55 seconds and 28 disabled nodes) and calculate the ratio of the acknowledgement packets to the whole packets traced in our simulation. Then we run our simulation in the presence of 4, 12, 20, 28, 36, and 44 disabled nodes separately with the other two parameters statically configured to typical values, and compare the six acknowledgement packets ratios.

Fig. 5a illustrates the packets traced in the typical scenario of our simulation. From the histogram, we can see that the SIP REGISTER packets account for almost 90% of SIP packets because the user agent should keep alive with its registrar after it has registered in the server, whereas the acknowledgement packets only accounts for about 5% of all packets traced in the simulation.



**Fig. 5.** a) The number of SIP packets and acknowledgement packets; b) Acknowledgement packets ratios for six different numbers of disabled nodes

Fig. 5 b) shows the ratios of acknowledgement packets to the whole packets in the presence of six different numbers of disabled nodes, and we can see that the ratio of acknowledgement packets changes very little, staying between 5.1% and 5.3%. The number of acknowledgement packets only depends on the number of the overall sessions and the number of the nodes which SIP messages traverse. In our simulation, the acknowledgement packets ratio can be maintained around 5.2%.

From the results of the two experiments above, we can conclude that our algorithm does not bring on much overhead on acknowledgement response packets no matter how many nodes fail or depart from P2P-SIP networks.

## 5 Related Work

In traditional SIP telephony networks, there are several works [2, 9] to achieve the reliability of SIP service. In [2], the authors have proposed a state-sharing algorithm to implement highly reliable SIP sessions by using reliable server pooling technique [4], but this method needs extra backup servers and consumes a significant amount of network bandwidth. In [9], the authors use IP takeover technique [12] to deal with the problem of SIP signaling transmission fault caused by the node failure, but IP takeover approach needs an extra heartbeat private network connection to monitor the health and status of each node in the cluster. Our approach achieves the reliability of SIP service without adding extra nodes and consuming extra network bandwidth.

IETF RELOAD [3], one of P2P-SIP protocols, achieves a simple reliable signaling transmission mechanism on the overlay link layer. Its basic principle is that the message sender will get an ACK from the receiver to detect the link or node failure, and the sender must implement a retransmission if timeouts occur. Our approach implements almost the same fault detection mechanism as the RELOAD protocol and the only difference between the two is that the fault detection mechanism is achieved on different layers. The failover routine of our approach is better than simple retransmissions since the retransmission mechanism only works when the link or node failure is temporary and our algorithm can transmit the message correctly even if the failure is permanent.

In P2P-SIP networks, several works [5, 10] have focused on the reliability of SIP registrations information, whereas our approach focuses on dealing with the problem of SIP signaling transmission fault and SIP transactions information loss.

## 6 Conclusion and Future Work

The existing P2P-SIP implementation has improved the scalability and reliability of tradition SIP telephony networks, but it corrupts the reliability of SIP service inevitably because the service nodes in large-scale P2P-SIP networks are probably more unstable than in traditional SIP networks. Our lightweight approach to address this problem simply replicates SIP transactions in the successors of the node which receives the SIP message to maintain the reliability of users' transactions information, and selects one the successors of the failed or departed node in P2P overlay as the takeover server to ensure the reliability of the SIP service. Our simulation results suggest that our algorithm maintains around 99% dialogs correctly in the presence of nodes failure or departure and provides the reliable SIP service in large-scale P2P-SIP networks with acceptable and reasonable overheads over the Internet.

We will implement the algorithm in our existing P2P-SIP demo system. By exactly checking the resource usage, call throughput, dialog completion ratio and algorithm overheads, we can explore the performance and the effectiveness of our algorithm in

the real system. Besides, our work can be extended to some other P2P-based signaling protocols to ensure the reliability of critical services.

## References

1. Baumgart, I.: P2PNS: A Secure Distributed Name Service for P2PSIP. In: Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communication, Hong Kong, China (March 2008)
2. Bozinovski, M., Renier, T., Schwefel, H., Prasad, R.: Transaction Consistency in Replicated SIP Call Control Systems. In: Proceedings of 4th International Conference on Information, Communication and Signal Processing, Singapore (December 2003)
3. Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., Schulzrinne, H.: REsource LOcation And Discovery (RELOAD) Base Protocol, IETF, draft-ietf-p2psip-base-08 (March 2010)
4. Lei, P., Ong, L., Tuexen, M., Dreibholz, T.: An Overview of Reliable Server Pooling Protocols, IETF, RFC5351 (September 2008)
5. Li, L., Zhang, C., Wang, Y., Ji, Y.: Reliable and Scalable DHT-based SIP Server Farm. In: Proceedings of the 51th IEEE Global Communication Conference, New Orleans, Louisiana (December 2008)
6. Maenpaa, J., Camarillo, G., Hautakorpi, J.: A Self-tuning Distributed Hash Table (DHT) for REsource LOcation And Discovery (RELOAD), IETF, draft-ietf-p2psip-self-tunning-01 (March 2010)
7. Prior, R.: Scalable Network Architectures Supporting Quality of Service, PhD thesis, Faculty of Sciences, University of Porto (2007)
8. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol, IETF, RFC 3261 (June 2002)
9. Singh, K., Schulzrinne, H.: Failover and Load Sharing in SIP Telephony. Technical Report CUCS-011-04, Columbia University, Computer Science Department, New York, USA (March 2004)
10. Singh, K., Schulzrinne, H.: Peer-to-Peer Internet Telephony using SIP. In: Proceedings of the 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Stevenson, Washington (June 2005)
11. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proceedings of ACM Annual Conference of the Special Interest Group on Data Communication, San Diego, California (August 2001)
12. High-Availability Linux Project, [http://www.linux-ha.org/wiki/Main\\_Page](http://www.linux-ha.org/wiki/Main_Page)
13. Network Simulator 2, [http://nslam.isi.edu/nslam/index.php/Main\\_Page](http://nslam.isi.edu/nslam/index.php/Main_Page)
14. Peer-to-peer SIP, [http://en.wikipedia.org/wiki/Peer-to-peer\\_SIP](http://en.wikipedia.org/wiki/Peer-to-peer_SIP)
15. P2PSIP status Pages, <http://tools.ietf.org/wg/p2psip/>

# A Load Balanced Two-Tier DHT with Improved Lookup Performance of Non-popular Data Items

Mayank Pandey and Banshi Dhar Chaudhary

Department of Computer Science and Engineering,  
Motilal Nehru National Institute of Technology, Allahabad, India  
{mayankpandey, bdc}@mnit.ac.in

**Abstract.** We have earlier proposed a two-tier hierarchical DHT architecture where nodes with higher uptime and more resources dynamically form the upper tier DHT ring. Each node of this ring acts as a super node for a DHT ring of lower tier nodes. The performance of lookups for non popular data items in this type of self organizing two-tier DHT is not as good as that of flat DHT due to the uneven distribution of ring sizes at lower tier. Also, this uneven distribution leads to unbalanced load at super nodes. In this paper, we propose a dynamic node migration algorithm for forming two-tier DHT which ensures even distribution of ring sizes at lower tier. Our proposed migration algorithm includes mechanisms for merging smaller rings and splitting bigger rings. Further, this even distribution of lower tier ring sizes leads to improved load balancing among super nodes. Simulation results indicate that this modified two-tier DHT architecture performs better than flat DHTs in terms of mean hop count to lookup non popular data items.

**Keywords:** Hierarchical DHT, Load balancing.

## 1 Introduction

Peer-to-peer (P2P) based overlay networks are popular for building large scale distributed applications due to their scalability, and self-organizing characteristics. They are broadly categorized as unstructured and structured p2p systems. Unstructured P2P systems, like Gnutella [1], use flooding techniques for searching data items whereas structured p2p systems, such as Chord [2] and Pastry [3], utilize Distributed hash table (DHT) to locate data items. These DHT based systems assume that participating nodes are homogenous in terms of computing resources and uptimes. They have high overhead of maintenance under high churn condition.

Hierarchical DHT based p2p systems [4] [5] [6] [7], have been proposed to utilize heterogeneity of nodes in terms of their uptime and quantum of computing resources to improve lookup and maintenance cost of flat DHTs. These proposals differ with each other in terms of overlay topology at different tiers of hierarchy.

We have earlier proposed a two-tier DHT architecture in [8] where the upper tier contains a DHT ring of nodes which have higher uptime and more computing resources. The nodes in this tier are referred as super nodes and they may dynamically join and leave the DHT ring. Each super node is logically connected to group of

nodes which themselves form a DHT ring. Such groups of nodes form the lower tier and have limited computing resources and are less stable in terms of their uptimes. The important property of this architecture is that the hashed ids of nodes in a lower tier group must lie between the hashed ids of super node and its predecessor. For the dynamic migration of nodes from lower tier to upper tier, the node at lower tier is required to send a request to its super node provided it has resources and uptime more than prescribed threshold. In this architecture, all requesting nodes which meet the threshold requirement are upgraded as super nodes. This upgrade leads to re-distribution of lower tier nodes amongst newly selected super nodes.

This scheme of node migration resulted in increased upper tier ring size and large number of unevenly distributed lower tier rings. As a consequence, it hampered lookup performance for non popular data items. This result prompted us to look for an improved strategy for node migration which ensures evenly distributed lower tier ring sizes.

In this paper, we present a super node migration algorithm which ensures even distribution of ring sizes at lower tier to improve the lookup performance of non popular data items. Further, we argue that even distribution of ring sizes at lower tier forms a load balanced upper tier of super nodes. Our migration algorithm includes procedures for:

- Splitting big rings and promoting nodes to upper tier as super nodes.
- Merging small rings and demoting super nodes to lower tier.

The rest of the paper is organized as follows. In Section2 we demonstrate the impact of uneven ring sizes at lower tier over the lookup performance. Further, the relationship between lower tier ring size and load experienced by associated super node is also discussed in this Section. In Section3 we present our improved super node migration algorithm. In Section4 we present the simulation environment and results. We discuss related works in Section5 followed by conclusion in Section6.

## 2 Lower Tier Ring Sizes

We have performed experiments to analyze the effects of migration and promotion algorithms of our earlier 2-tier DHT [8]. We have simulated the joining of nodes at lower tier and migration of prospective super nodes to upper tier. This simulation have been performed over approximately  $2^{16}$  nodes (number of hosts in Class B of IP address), where some percentage of nodes are designated as prospective super nodes in random manner. We have measured the lower tier ring sizes and divided them in four classes. This classification depends upon the value of  $K$  which is defined as  $N/S$  where  $N$  is total number of nodes and  $S$  is number of super nodes.

- Perfect Ring (size =  $K$ )
- Big ring (size  $\geq 2K$ )
- Small ring (size  $\leq K/2$ )
- Reasonable ring (  $K/2 \leq \text{size} \leq 2K$ )

It may be noted that as per above classification, *perfect rings* are subset of reasonable rings.

Table.1 summarizes the results of our simulation experiments. It may be observed that on an average 40-45 percent of rings are small whereas percentage of perfect rings is very less. Average percentage of big rings is 15 and reasonable rings are 40-47 percent.

**Table 1.** Variable Lower tier Ring Sizes: Results of Experimental Analysis

Super Node %	Small Ring %	Reasonable Ring %	Big Ring %	Perfect Ring %
5	39.01	47.92	13.07	2.60
2	38.21	48.51	13.28	1.27
1	39.19	47.30	13.51	0.60
0.5	40.15	44.53	15.33	0.73
0.1	40.24	47.56	12.20	1.20
0.05	47.50	35.00	17.50	2.50

Even though the reasonable ring have been defined on the basis of ring sizes between  $K/2$  to  $2K$ , a finer observation of our experimental results also revealed that a large percentage (almost 60 to 65 %) of these reasonable rings have sizes nearly equal to  $2K$ . Taking this into account along with the percentage of big rings, it becomes obvious that the rings at lower tier are either very small ( $<K/2$ ) or very large ( $\geq 2K$ ) and the average percentage of perfect rings (size equal to  $K$ ) is only order of 2-3%. This uneven distribution of ring sizes increases mean hop count for lookups, as demonstrated in the simulation results of [8].

To further analyze the impact of uneven ring sizes at lower tier, we need to look in detail the lookup cost of 2-tier DHT. To lookup a key in 2-tier DHT, a node at lower tier sends lookup request to its super node. The responsible super node is found by using standard Chord lookup procedure in upper tier. This lookup procedure takes  $\frac{1}{2} \log S$  hops on an average where  $S$  is the number of super nodes (proof is provided in [2]). Responsible super node then routes the lookup request in its lower tier ring and it takes  $\frac{1}{2} \log L$  hops on average to find the target node. Here  $L$  is the size of lower tier ring associated with responsible super node. Thus, the average number of hops for looking up any key in 2-Tier DHT is:

$$\text{Lookup2Tier} = \frac{1}{2} \log S + \frac{1}{2} \log L \quad (1)$$

From (1) it is evident that the proposed 2-tier DHT cannot have better lookup for non popular data items than a single tier DHT. When all rings are perfect ( $L=K=N/S$ ), the number of hops for lookups in 2-tier DHT is equal to  $\frac{1}{2} \log N$ , same as single tier Chord DHT of  $N$  nodes. Even for two extreme situations, where the upper tier DHT ring contains either only one node or all the nodes, the lookup performance of 2-tier DHT is same as that of single tier Chord DHT. These situations are unrealistic in the light of assumed heterogeneity of nodes in terms of computing resources and uptimes.

This limiting lookup performance is compensated by lower maintenance cost in comparison with flat DHT. Further, skew-ness in the request pattern is utilized to cache the popular data items at super nodes. This improves the lookup performance of

2-tier DHT for popular data items in comparison with flat DHTs. These two features are compelling reasons for designing our earlier 2-tier DHT architecture.

All the above analysis encourages us to look for a design strategy which avoids the two extreme situations mentioned above and ensures even sized rings at lower tier to achieve better lookup performance for non popular data items.

Further, the load experienced by a super node in 2-tier DHT is dependent on the size of associated lower tier ring. As the ring size increases, super node becomes responsible for bigger range of id space and consequently more queries are forwarded towards it. Secondly, it has to handle more number of super node stabilization queries and lower tier finger table updates. In the case of popular data items, super node has to manage bigger cache and has to handle more number of routing requests both from lower and upper tier rings. Thus, the number of nodes in lower tier rings should be evenly distributed for better load balancing among super nodes.

### 3 Algorithms for Even-Sized Rings at Lower Tier

The dynamics of changes in the sizes of lower tier rings are dependent on joining and departure of nodes, migration of nodes to upper tier and departure/failure of super node. The migration of nodes from lower tier to upper tier results in split of lower tier rings whereas departure or failure of super node leads to merging of rings. The splitting and merging of rings are controlled by two limiting thresholds:  $Split = 2K$  and  $Merge = K/2$ . Where  $K = N/S$ ;  $N$  is the total number of nodes and  $S$  is the number of super nodes.

*Split* represents a hard limit, at which the lower tier ring is divided into two rings and a lower tier node is promoted to upper tier as super node. *Merge* also represents a hard limit, at which the lower tier ring is merged with the ring attached with successor super node and previous super node is demoted to lower tier.

Nodes in lower tier of 2-tier DHT architecture, send periodic stabilize message to their super-node and hence every super-node can know the count of normal nodes in its lower tier ring. In this manner, super node periodically monitors the lower ring size and takes decision accordingly whether to merge or split the ring.

#### 3.1 Migration Algorithms

In this sub-section we present our migration algorithm to evenly distribute the number of lower tier nodes among super nodes. Two variables are maintained at super node for our migration algorithm. There is a *Strong node list* which lists all the nodes who have requested to become super node and a variable *Attached ring size* which contains the count of number of normal nodes under a super node.

Nodes in lower tier which are capable to become super nodes send an *upgrade\_level\_request* to their current super node. Instead of sending an *upgrade\_level\_response* instantly, a lazy promotion policy is followed. As per this policy a super node only keeps track of these requesting prospective super nodes. These nodes are termed as strong nodes and their list is maintained at current super node. After each super node stabilization period, the *attached ring size* is inspected. If the ring size is greater than or equal to  $2K$  then the ring is split into two. Meanwhile, the

*strong node list* is also getting populated as lower tier nodes are requesting to become super node. From this list, a node which can provide most even split is chosen to become a super node. This chosen strong node gets an *upgrade\_level\_response* from current super node and it migrates from lower tier to upper tier according to the *Split()* algorithm of our earlier 2-tier DHT [8]. Further, if the *attached ring size* is less than or equal to  $K/2$  then ring is merged with the ring under successor super node. The super node of merged ring is demoted to lower tier is listed in *Strong node list* of current super node. Fig.1 gives the modified promotion/demotion algorithm. The *Merge()* procedure is described in Fig. 2. This procedure is made keeping in mind the property of 2-tier DHT architecture which suggests that id of every lower tier ring node should be between super-node's predecessor id and super-node's id.

1. *Nodes want to become super nodes*
  - a. *send requests to its current super node*
  - b. *Strong node list is updated*
2. *After every periodic super node stabilization:*
3. **if** (*size of ring  $\geq 2K$* )
  - a. *Chose best alternative from Strong list*
  - b. *Promote it to become super node*
  - c. *Send upgrade\_level response*
  - d. *Split and Migrate // call **Split()***
4. **else if** (*size of ring  $\leq K/2$* )
  - a. *Demote the super node to lower tier*
  - b. *Merge the ring // call **Merge()***
  - c. *Make entry of demoted super node as strong node*

**Fig. 1.** Modified Promotion/Demotion Algorithm

- *Suppose A and B are super nodes.*
  - *A is successor of B*
  - *Ring under B is getting merged with ring under A*
1. *A's upper tier predecessor is updated to B's upper tier predecessor*
  2. *A's lower tier successor is updated to B's lower tier successor*
  3. *B's lower tier successor is updated to A's lower tier successor.*
  4. *B's upper tier predecessor link is freed.*
  5. *Predecessor of B's lower tier successor is updated to A*
  6. *Predecessor of A's lower tier successor is updated to B*

**Fig. 2.** Merge Algorithm

The procedures described above depend on the value of  $K$ . In a very large evolving network where number of nodes keeps on changing dynamically, it is not possible to calculate the value of  $N$  and  $S$  and consequently  $K$ . Thus, we require a mechanism where super nodes are able to estimate the value of  $K$ . As already mentioned, normal



nodes send stabilize message to its super-node periodically and hence every super-node can know the count of normal nodes in its lower tier ring. We consider this count as a  $K$  value for a super node. These  $K$  values are exchanged among super nodes by piggybacking them over finger table and successor list update messages. These update messages are sent periodically by every super node to maintain the structure of upper tier ring. In this way, super nodes can accumulate enough samples to find out the mean  $K$  value at any given time.

## 4 Simulation Environments and Results

We have simulated our modified design by using Oversim [9] above OMNeT++ [10]. OMNeT++ is an open-source, component-based, discrete event simulation environment coded in C++. OverSim is an open-source simulation framework for peer to peer overlay networks to be used over OMNeT++ simulation environment. The simulator contains several modules for structured p2p systems e.g. Chord, Kademlia and Pastry. We extended its Chord module for simulating our modified 2-tier DHT architecture.

We performed simulations for single tier Chord and our modified 2-tier Chord to compare the mean hop count required for routing messages. The message is created as a data packet which contains random key value generated periodically at every node. It also contains a *hop\_count* field which gets incremented every time the data packet arrives at an intermediate node. UDP is used as transport layer protocol for transmitting messages to other hosts. A *global\_observer* keeps a list of known nodes already connected to network. When a new node joins, it works as a bootstrap node and fetches a random node from the list for initializing joining process.

We simulated two types of nodes; normal and super nodes. Rate of churn of nodes is kept higher for normal nodes and comparatively lower for super nodes. Oversim framework is initialized by *.ini* file. This file contains several network parameters like number of nodes, churn rate and the ratio of super node to normal nodes and are varied dynamically at runtime. Other parameters like delay between finger updates, time period for successor list update and migration delay etc. are also initialized from *.ini* file. Results are obtained for a simulation time of twenty four hours by using the statistics class of OMNET++.

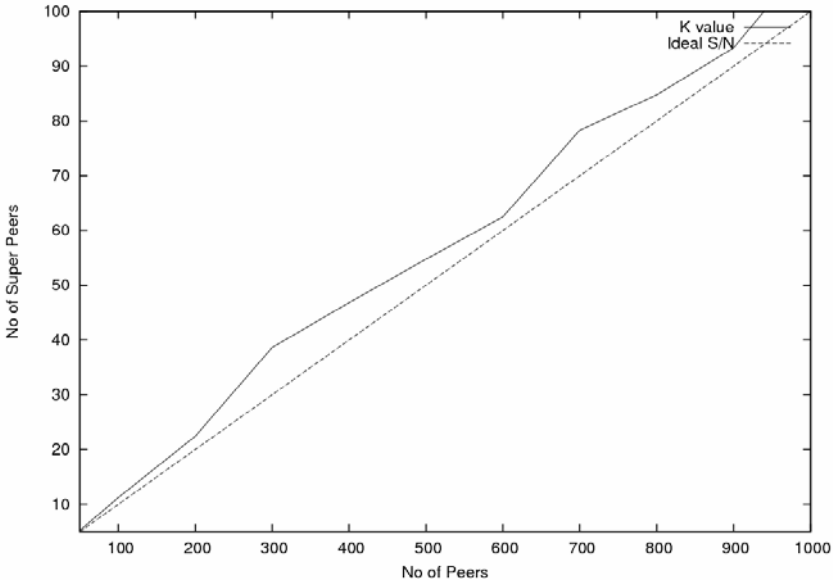
We captured several snapshots of our architecture during execution of simulation to evaluate the performance of our modified migration algorithms in terms of lower tier ring sizes. During simulation, number of nodes at upper and lower tier constantly changes as nodes join and leave according to specified churn rate. Table.2 provides the consolidated results gathered from these snapshots which are acquired for different percentages of nodes capable to become super nodes in system.

From Table.2 it may be observed that on an average approximately 80% of rings are evenly distributed (ring sizes around  $K$ ). Big rings (size around  $2K$ ) are in the order of 10-11 % whereas small rings (size around  $K/2$ ) are about 8-10%. This suggests the applicability of our modified migration algorithm to produce evenly distributed ring sizes at lower tier.

Fig.3 depicts the comparison between estimated  $K$  value and ideal  $N/S$  ratio. Results are collected for different values of total number of nodes and super nodes represented on x and y axis of graph respectively and 10% of nodes are considered in the role of super nodes. It may be observed that there are no unexpected oscillations in estimated  $K$  values, though they are slightly greater than ideal ratio.

**Table 2.** Consolidated Results of Snapshots

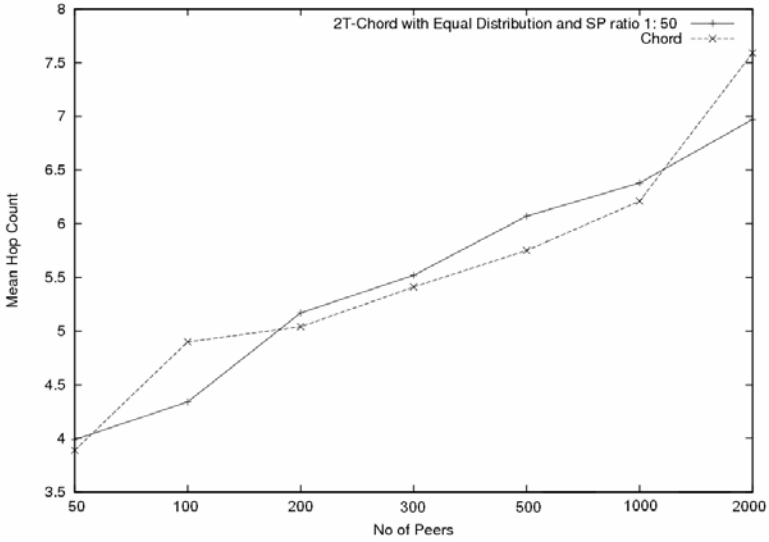
SN %	Size around K/2	Size around K	Size around 2K
5	9.16	78.29	10.34
2	8.23	82.17	10.08
1	9.07	81.25	10.43
0.5	10.34	79.46	11.21



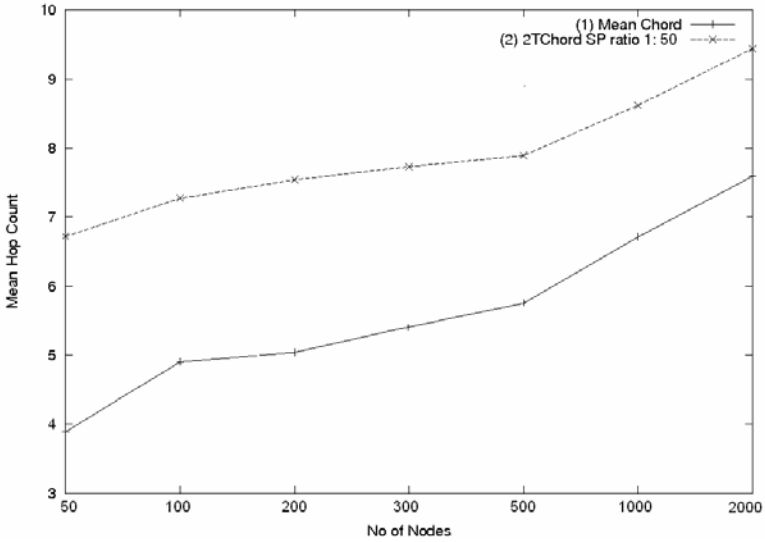
**Fig. 3.** Estimated K Values

Fig.4 depicts the relationship between mean hop count for lookups and number of nodes for our modified design. Fig.5 depicts the lookup performance for our earlier design of 2-tier DHT where even distribution of lower tier nodes across super nodes is not attempted. In Fig.4 and 5, results are collected for 50, 100, 200, 300, 500, 1000 and 2000 nodes shown on the x-axis of graphs. These results are obtained for 1:50 super node to normal node ratio.

By inspecting Fig. 4 and 5 together, it becomes obvious that the mean hop count in our modified design is considerably lower than the earlier design for the same super-node to normal node ratio. In Fig.4, dotted curve and solid curve depicts mean hop count in Chord and in our modified design respectively. It may be observed that on an average, performance of our modified design is nearly similar to Chord. In Fig.5, solid and dotted curve represents the mean hop count for lookups in single tier Chord and our earlier 2-tier DHT respectively. It may be noted that in earlier 2-tier DHT the mean hop count for lookups is much higher compared to single-tier Chord.



**Fig. 4.** Lookup Performance of Modified 2-tier DHT



**Fig. 5.** Lookup Performance of Earlier 2-tier DHT

It is significant to note that our earlier design of 2-tier DHT performs well in comparison to single tier Chord when lookup is made for popular data items cached at super nodes. Also, the lookup performance increases with the increasing probability of finding data items at super nodes as shown in the simulation results of [8]. Further, the earlier design performs better than single tier Chord in terms of maintenance cost.

Our modified design inherits all the good characteristics of earlier design like reduced maintenance cost and better lookup performance for popular data items. Further, this design ensures that if there is very limited skew-ness in request pattern i.e. the lookup is made for non-popular data items, the performance of our design is at least as good as single-tier Chord.

## 5 Related Work

Several research efforts [4][5][6][7], have been made for designing hierarchical DHT based peer to peer overlays. These efforts differ with each other in terms of overlay topology followed at various layers of hierarchical design. Further, these efforts take into account different tradeoffs for designing such architectures, for e.g. maintenance cost versus lookup cost, similar identifier space for all the layers versus separate identifier space for different layers, random joining of lower tier nodes in a cluster versus cluster of topologically close nodes etc.

Some research efforts [11] [12] [13], studied the potential benefits of hierarchical DHTs over their flat counterparts. In [11], an analytical study is performed to analyze the lookup cost of hierarchical design where lower tier peers are organized in groups based on their topological closeness. The main focus of this analysis is the impact of the churn rate of upper and lower tier nodes over the lookup up performance of hierarchical DHT. It is argued that lookup performance of hierarchical design is improved under certain churn conditions. In [12], an analytical cost framework is proposed to evaluate the flat design, in which all nodes are assumed to be homogenous against the super-peer design, in which a small subset of peers interconnect groups of topologically close nodes with less uptimes. The results demonstrate that no design can be considered better universally as the advantages and disadvantages of a design depend on multiple alternatives for a specific application.

An approach closer to ours is presented in [13]. In this approach, an analysis of the maintenance and lookup cost of hierarchical DHT structure is done. The analysis is performed for a design where group of lower tier do not maintain any structure and are directly connected to their super peer. Further, a load balancing approach at upper tier is also presented where the number of leaf nodes attached to a super node, is considered to be load of super node. In our approach, lower tier groups also form DHT ring and it has been demonstrated that uneven distribution of lower tier ring sizes not only affects load balancing of upper tier, it also affects the routing performance of non popular data items.

## 6 Conclusion

We have presented a dynamic node migration for a two-tier DHT architecture which ensures even distribution of ring sizes at lower tier. We have demonstrated that that our strategy improves the routing performance when lookup is made for non popular data items. This even distribution resulted in better load balancing among super nodes at upper tier. Further, it also inherits lower maintenance cost and better lookup performance for popular data items from earlier design of our two-tier DHT architecture.

## References

1. Gnutella2 Website, <http://www.gnutella2.com>
2. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: ACM SIGCOMM (2001)
3. Rowstron, A., Kermarrec, A.-M., Castro, M., Druschel, P.: SCRIBE: The design of a large-scale event notification infrastructure. In: Crowcroft, J., Hofmann, M. (eds.) NGC 2001. LNCS, vol. 2233, p. 30. Springer, Heidelberg (2001)
4. Ganesan, P., Gummadi, K., Garcia-Molina, H.: Canon in G Major: Designing DHTs with Hierarchical Structure. In: International Conference on Distributed Computing Systems (ICDCS 2004) (2004)
5. Li, Y., Huang, X., Ma, F.-Y., Zou, F.: Building Efficient Super-Peer Overlay Network for DHT Systems. In: Zhuge, H., Fox, G.C. (eds.) GCC 2005. LNCS, vol. 3795, pp. 787–798. Springer, Heidelberg (2005)
6. Peng, Z., Duan, Z., Qi, J.-J., Cao, Y., Ertao, L.v.: HP2P: A Hybrid Hierarchical P2P Network. In: First International Conference on the Digital Society (ICDS 2007) (2007)
7. Zhao, B.Y., Duan, Y., Huang, L., Joseph, A.D., Kubiatowicz, J.D.: Brocade: Landmark routing on overlay networks. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, p. 34. Springer, Heidelberg (2002)
8. Pandey, M., Ahmed, S.M., Chaudhary, B.D.: 2T-DHT: A Two Tier DHT for Implementing Publish/Subscribe. In: 12th IEEE International Conference on Computational Science and Engineering, Vancouver (2009)
9. Oversim Web Site, <http://www.oversim.org/>
10. OMNET++ Web Site, <http://www.omnetpp.org/>
11. Garcés-Erice, L., Biersack, E., Ross, K.W., Felber, P.A., Urvoy- Keller, G.: Hierarchical P2P Systems. In: ACM/IFIP Conference on Parallel and Distributed Computing (2003)
12. Artigas, M.S., López, P.G., Skarmeta, A.F.: A Comparative Study of Hierarchical DHT Systems. In: 32nd IEEE Conference on Local Computer Networks (LCN 2007), Dublin, Ireland (October 2007)
13. Zoels, S., Despotovic, Z., Kellerer, W.: Cost-Based Analysis of Hierarchical DHT Design. In: Sixth IEEE International Conference on Peer-to-Peer Computing (P2P 2006), Cambridge, UK (September 2006)

# Gridlet Economics: Resource Management Models and Policies for Cycle-Sharing Systems

Pedro Oliveira, Paulo Ferreira, and Luís Veiga

INESC-ID / Instituto Superior Técnico - Technical University of Lisbon  
Rua Alves Redol No 9, 1000 Lisboa, Portugal  
pedro.f.oliveira@ist.utl.pt, {paulo.ferreira,  
luis.veiga}@inesc-id.pt

**Abstract.** In cycle-sharing peer-to-peer systems, the users contribute to a pool of resources which they can all use. The access to the resources can be made by many users simultaneously, so there is the need to define which resource each one will use. In this paper we propose an economic model for the management of resources in those systems, matching jobs to resources according to a flexible set of requirements. In order to use the resources of the system the user makes a transaction where he exchanges credits for the right to use them, those credits can only be received by previously contributing to the system. Thus the model encourages or forces the users to contribute, which is essential in a peer-to-peer system. To reduce the risk of the transactions a reputation system is used that penalizes misbehaving users.

**Keywords:** Cycle-sharing system, peer-to-peer network, resource management, economic model.

## 1 Introduction

In peer-to-peer cycle sharing systems the users make their home computers, and respective computational resources, available to be used by others. This way they create a global pool of resources with a huge computational power. The idea is that when they have a job that is very computationally intensive and that would take too long to be executed only on the machines that the users own, they can use the resources of the pool to make execution much faster. However, there is the need to regulate and manage the use of the resources in that pool. In this paper we propose a new model designed to do that management, called Gridlet Economics.

Although much research has been done already on peer-to-peer systems [1], there is the need for a new model because so far the main focus has been on file-sharing. The resource management model used in those systems only deals with a binary requirement, having the file or not. This cannot be applied to cycle sharing, because these environments have to deal with many and varied requirements, such as CPU speed, number of cores or OS installed. Nevertheless, other resource management models can be considered, such as the ones used in the BOINC system or in Grids. Despite not being intended as peer-to-peer, these systems also use several commodity computers to create a huge pool of resources. However, the strict client server

architecture used in BOINC does not consider concurrency to the access of resources, since the only one that can use them is the central server hosting the project. This is unusable in a peer-to-peer system since one of the main characteristics of those systems is that everyone can act as a client and a server. On the other hand, in Grids, it is assumed that all computers are trustworthy, that the components are relatively static and that there is no need to encourage contributions, assumptions that cannot be made in a peer-to-peer system.

Therefore, in this paper we present Gridlet Economics, an economic model for the management of the resources in a peer-to-peer cycle sharing system which is able to match the jobs to the resources according to a rich set of requirements. The basis of this model are the transactions where a user exchanges credits for the right of using the resources of the system. Those credits can only be acquired by contributing to the system, thus the model drives the users to contribute, which is essential in a peer-to-peer system because it depends on the contribution of the participants to survive. Simultaneously, this model prevents users from controlling too many resources and/or their prices by virtue of their accumulated credit, keeping the system fair. Moreover it regulates in a fair manner the access to the resources, because users which contribute more have more credits and therefore can use more of the resources of the system. Also, this model uses a reputation system to penalize misbehaving users and prevent them from benefiting from their actions.

## 2 Related Work

**Economic Models:** When there is demand and supply of resources there has to be some management of who uses what and when. In the real world, economic models have been used to do so for centuries and, with the appropriate regulation and regardless of ideology, have proved to be a successful and sustainable way of governing the exchange of resources, goods and services. Also, the use of an economic model provides a scalable option for the management of resources, especially when they are not all in the possession of the same entity because each user regulates the use of its own resources. Moreover, it provides a simple method for defining the priority order of the jobs, by establishing that the ones for which the users are willing to pay more have the highest priority. Another advantage of the use of an economic model is flexibility since it allows a uniform treatment of all kinds of resources, from CPU time to application version. In fact, there are already some systems that use economic models for resource management [2, 3]. However, none addresses the particular aspects of resource management in a peer-to-peer cycle-sharing system.

When using an economic model there are three main aspects that have to be considered: “the currency”, “how the price is defined”, by the supplier or seller, and “how it is selected”, by the consumer or buyer. The selection of what is used as currency is an important factor because it is what is exchanged in the transactions. The currency-based systems can be divided into two main categories: non-monetary and monetary. The non-monetary systems, such as bartering [4] and direct exchange are used in primitive economies because they are simple and easy to build, however they are very inflexible and limited. On the other hand, monetary systems are much more flexible, but also more complex due to the use of money. Virtual money is one form of money

that only has value inside the system in which is used. This type of money, used in [16], is considered appropriate for systems without a controlling third party, such as the P2P system, because it has far fewer security concerns than real money.

One of the bases in an economic model is that the consumer has many options of services, or resources, provided by different suppliers, from which he can choose. Each alternative offers a different type of service for different cost and a decision of which is selected has to be made. In order to make that decision, a function that calculates the utility value of each option is used. The utility is a measure of the relative satisfaction of the consumer with the consumption or purchase of a determined good or service, so the right selection can be made by choosing the option that maximizes that value. On the other hand there is also the need to define the price that the suppliers will charge for the resource consumption, or use, and to do so there are several ways. The more traditional one is Commodity market where the resource owners specify the price and the consumers only have the choice of buying or not. The prices can be defined by the owner using a flat policy, where the price does not change for a certain amount of time, or variable one, where price changes very often based on the amount of supply and demand. Other options exist, such as bargaining or auctions [11], where the consumers not only have the choice of buying or not, but also influence the value asked. It is up to the resource owner to decide which method is used.

**Reputation:** Trading involves always a certain amount of asymmetric risk (that one of the parts will not fulfill its obligations once the other has committed the resources or currency) and, if it is not controlled, that risk can lead to the collapse of the economic model. The reputation is a good way of reducing the risk without the need of a supervising third party. Besides reducing the risk, reputation can also be used as a mechanism to induce good behavior in this type of markets. That concept is not new, in fact several economists have already published work analyzing its proprieties and one of the major auction sites in the world, eBay [6], relies almost exclusive on a system like this to reduce the risk and induce good behavior on the part of its members.

In order to calculate a user's reputation the system collects the classification that the users give to him and then converts it into a value that can be quantified and compared using a reputation function [5]. In a centralized system, like eBay, that value is calculated and stored in a central server which is considered a trusted third party. However there are decentralized alternatives such as the Eigen Trust Algorithm [7], which can be used on a peer-to-peer system.

**Resource discovery:** As it has already been said, one of the basis of an economic model is the possibility of selecting the option that maximizes the utility of the resource consumer. In order to make the selection there is the need to know what options are available, for that there is the need to discover the resources available at a certain time. Regarding resource discovery, peer-to-peer systems are divided into three main categories: unstructured, structured and hybrid.

In unstructured peer-to-peer systems, such as Gnutella, the nodes are randomly distributed and therefore have low costs to enter and leave the system, which makes them appropriate for highly-transient populations. However, because there is no information about the organization of the peers, these systems have to use uninformed search techniques which are very inefficient. To compensate for this, Kazaa uses a hierarchic approach with two classes of nodes, super peers and ordinary nodes.



Only the super peers participate in the search, this way the number of nodes searched is greatly reduced, making it more efficient.

In structured peer-to-peer systems such as Chord [14], CAN (Content Addressable Network) [10], Pastry [12] and Kademia [8], the peers are distributed using a DHT (Distributed Hash Table) in which peers and keys are mapped through the hash function. This allows the finding of the peer corresponding to a determined key to be very efficient. However, the high cost for entering and leaving this type of systems makes them less suitable for highly transient populations.

Many hybrid systems try to maintain the efficiency of the search of the structured systems while also employing a less strict organization, akin to unstructured system, thus making them more suitable for transient populations. This is usually achieved by having a set of super-peers belonging to a smaller structured overlay, while the majority of other peers discover content and resources only by mediation of a super-peer.

### 3 Gridlet Economics

The model proposed is designed to be applied on top of a peer-to-peer cycle-sharing system. In this system the users make their computers, and respective computational resources, available to be used by the other participants. When a user wants, he can submit jobs that are executed on the resources of the other users of the system, making its execution much faster. In order to make it possible for the job to be distributed over the resources available in the system, it is split into smaller work units. These units, called gridlets, are the main input of the model and it is assumed that an existing layer is responsible for their creation and the aggregation of the result of their execution, such as that described in [15].

In practice, the objective of the model is to offer a decentralized mechanism that maps the gridlets submitted to the computers where they will be executed. The selection must take into consideration a set of requirements that correspond to the user utility and the contribution that the user has made to the system. These requirements will be represented and evaluated using the method described in Partial Utility Algebra [13].

**Model Overview:** In this model the computers present in the system, called the nodes, can be classified as producers, if they are contributing with resources to the system where gridlets can be executed, or consumers, if they have gridlets being executed in the system and therefore are consuming the resources of the system. Although a node can have both classifications at the same time, normally it will alternate between them, since when it is a consumer it will not contribute with his resources to the system, but use them to also execute some of its gridlets. The model also uses a hierarchical approach with two classes of nodes, brokers (similar to super peers) and ordinary nodes. The brokers maintain a global distributed index where all the nodes that are contributing to the system are registered and use it to select the executor of a gridlet, i.e. the producer that will execute it. Although the classification of the nodes is independent of the division in classes, it is considered that the brokers will not be producers and therefore will not execute gridlets. This is done in this way because if a broker had to simultaneously execute a gridlet and make the selection of an executor, both activities would be slower, therefore, since the selection of executor is a crucial

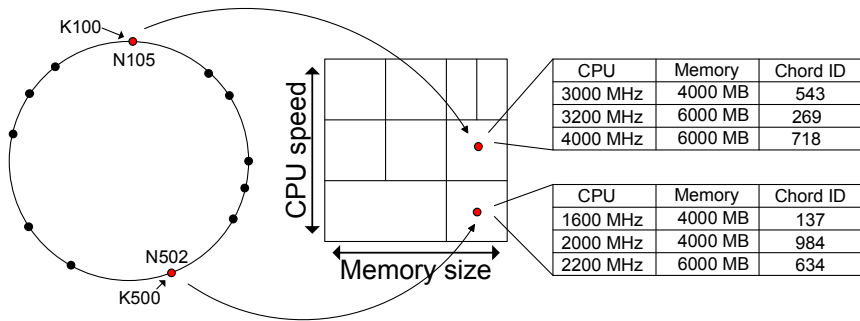
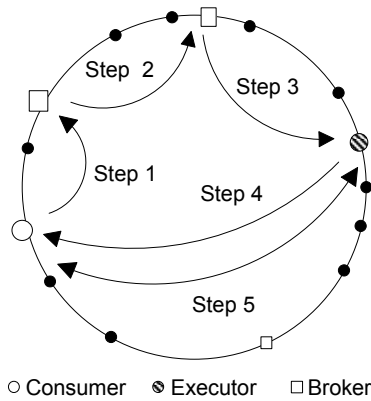


Fig. 1. Model overview

element of the model precedence is given to that task. Besides, when the brokers are selecting a producer to execute a gridlet, they are already contributing with resources to the system. Also, a possible conflict of interests could arise, similar to inside trading, because that node could be at the same time, the selector and the selected for the gridlet execution.

All the nodes in the system will be distributed over a Chord ring. The use of this look up service provides an efficient communication mechanism and a unique ID for each node that can be maintained between sessions. Also, the brokers are defined as the nodes that are responsible for a predefined set of keys (e.g. 1000, 2000, etc.) in the ring. This way when a node wants to communicate with a broker, he only has to calculate the closest predefined key and send the message to it. This allows a transparent change of the node that is acting as the broker and an even distribution of the load across all them. Nevertheless, all of brokers can be easily found, which means that if the user wants he can choose to use another one. Figure 1, shows an overview of the model, where the nodes N105 and N502 are responsible for the pre-defined keys K100 and K500 in the Chord ring and therefore are brokers, which maintain a distributed index of the nodes in the system.

The bases of this economic model are the transactions, where the consumer pays to have its gridlets executed on the resources available in the system. The payments will be made using credits, a currency of a virtual money system. The execution of each gridlet is considered a different transaction and it starts when the gridlet is received by the consumer (depicted next in Figure 2). First, it attaches to the gridlet the requirements that represent the user utility and then sends it to a broker (Step 1). Next, the broker that receives the gridlet, together with the other brokers, selects the node that according to the specification is better suited to execute it (Step 2). Then the gridlet is passed to the node selected and if that node is idle it will immediately execute the gridlet, otherwise it will put it in a local queue to be executed later (Step 3). During the execution, the producer executing the gridlet will monitor its resources' consumption and, after the execution terminates, its result is sent back to the consumer along with an invoice (Step 4). In the end, the consumer pays the invoice, which includes the nodes that stored the result, and classifies the producer. After receiving the payment the producer classifies the consumer, thus ending the transaction (Step 5).



**Fig. 2.** Steps in a transaction

**The Market Square:** The market square is an index where the producers advertise their resources, so that they can be selected to execute the gridlets of the consumers. In practice, it is a table that contains an entry for all of the nodes that are contributing with their resources to the system. Each entry will have the Chord ID of the node and its characteristics (price, CPU speed, memory size, OS installed, etc.). This table will be indexed by a set of predefined characteristics, called the unit of cost. Since the search is done primarily through those characteristics, they should be things in which most users have an interest in, such as price, CPU speed, memory size or network bandwidth.

However, in a peer-to-peer system it is not possible for one node to store information about all the other nodes in the network, so the index will have to be distributed among all the brokers. In order to distribute the index among them without losing the ability to efficiently search through it, the brokers will organize themselves in a d-dimensional CAN, where each dimension corresponds to a characteristic of the unit of cost. Then, each broker will have the part of the index which corresponds to the characteristics that match the zone for which it is responsible. This means that the node responsible for the zone between 1950 and 2050 in the dimension of the CPU speed will have the index of the ordinary nodes that have the CPU speed in that range. In this way, it is still possible to travel efficiently through the index using the CAN routing mechanism. Though, normally the range of values and scale will differ from characteristic to characteristic and, if those exact values are used to define the CAN dimensions, this would hinder its routing system. So, instead of using the actual values of the dimensions a canonical representation is used. For example, from 0 to 100 and the conversion is made by calculating a percentage against the difference between the bottom and top value.

**Selection of the Executor:** The selection of the node that will execute the gridlet is done by evaluating the entries of the market square according to the requirements specified in the gridlet, which represent the user utility, and then choosing the node that is better suited to execute it. This means that all the nodes contributing to the system have to be published in the index of the market square, otherwise they will never be selected to execute a gridlet and therefore it is as if they were not contributing.

However, as the number of entries of the index increases, it will become unfeasible to evaluate them all. Also, in real economy this sometimes happens, where consumers are unable to check the individual price of all producers or sellers. Furthermore, the index is split into zones that are stored in different nodes. So, instead of doing a full index search, the search starts at the point that is the most likely to maximize the user utility and stops when it estimates it that has reached the best alternative.

To find the point that is the most likely to maximize the user utility, the unit of cost is used. For that, it is assumed that those characteristics have a great interest to all of the users and also that they all agree in whether they should be maximized or minimized. In an economic cycle-sharing environment this can be easily achieved, if the characteristics considered are features such as price, processor speed or network bandwidth. Thus, the broker that first receives the gridlet uses the routing of CAN to send it to the point that maximizes the utility, which corresponds to the maximum / minimum limit of the dimensions of the characteristics of the unit of cost that should be maximized or minimized, respectively. When the gridlet reaches the best suitable point, a local search is done, which is made by evaluating the nodes in that part of the index and determining the best one. Then an estimation of the values present in the adjacent zones is evaluated. The adjacent zones correspond to the neighbors in CAN, so this is called the neighbor search. This is done to estimate the classification that could be achieved if a local search was done in one of those brokers. If the classification of the node selected locally is higher than the one of the best neighbor, the gridlet is sent to that node to be executed. Otherwise, the gridlet is sent to the broker responsible for best adjacent zone, which repeats the same search steps. The search stops when a node is selected, or neither the node selection nor the neighbor selection returns a possible result; in that case the search fails and the gridlet is returned to the node that submitted it to the system. Figure 3 shows the pseudo-code for the search in the index.

The search is directed in a contrary direction to the one indicated in the unit of cost, i.e. the search can only travel down in the characteristics that are supposed to be maximized and up in the ones that are minimized. This means that only the neighbors

```

// search in one zone of the index
searchTheIndex ( gridlet ) {

    node    = localSearch ( gridlet.specifications );
    neighbor = neighborSearch ( gridlet.specifications );

    // both searches failed
    if( node.classification < 0 && neighbor.classification < 0 )
        return fail;

    if(node.classification >= neighbor.classification)
        node.execute ( gridlet );
    else
        neighbor.searchTheIndex ( gridlet );
}

```

**Fig. 3.** Pseudo-code for the search in the index

that respect this rule are considered in the neighbor search. That restriction is applied to avoid loops without requiring the brokers to maintain state of the ongoing searches, which is a great advantage when considering a great number of simultaneous searches, because it significantly improves the scalability and reliability.

**Prices and payments:** The consumers pay for the execution of their gridlets and this creates the need to define how much they will pay. However, the consumption of resources that occurs during an execution can vary much from gridlet to gridlet and it is difficult to estimate beforehand how much resource consumption there will be. This means that if the total price charged for the execution was determined beforehand, one of the parties would most likely lose with that transaction. So, instead of determining the price for the entire execution the producer defines a fee, which is the value that will be multiplied by the resources used (in essence, the unitary price charged). This fee can be the same for all the resources or be specified for each resource in particular. The definition of the value of the fee is determined as it is done in the Commodity market, which means that the producer is the only responsible for its definition and will then publish it in the index. This way, the fee is incorporated in the selection of the executor and selected according to the consumer utility. As an incentive to contribution in times of great demand, a variable price definition policy is used, where the price is altered based on the amount of supply and demand. The decision of raising or lowering the fee asked is made automatically based on the size of the queue of gridlets waiting to be executed on that node and on the nodes that have similar characteristics. That information can easily be supplied by the broker that stores the node index entry, since the other nodes will also have the entries in the same zone.

The brokers are also contributing to the system, however only the execution of gridlets is paid, which means that they are not rewarded for their contribution. So, in order to compensate them, they charge a tax for each transaction in which they participate. However, due to the high number of brokers that participate in a search, it is not feasible to pay a percentage to all them. Because for that to be done the taxes would have to be very high or the value paid to each one would not be relevant. So, only the first and the last broker involved in each search are paid. In this way, since the selection of the first broker done by the consumer is random, half of the taxes will be distributed uniformly among the all brokers. And the other half will go to the brokers that do the neighbor and local search, which are the operations that require more work.

**Risk and reputation:** In every transaction there is the risk that one of the parts might not behave properly and tries to take advantage of the other. In this model this can also happen, so a reputation system is used to minimize and control that risk by penalizing the users that act maliciously.

As a producer participating in a transaction, the user can take advantage of the consumer in two ways: fraud and overpricing. It is considered fraud when the producer instead of executing the gridlet returns a fake result and asks to be paid. When this happens, and is detected [9], the consumer gives a bad classification to the transaction and refuses to pay.

Overpricing is when a producer claims that the execution used more resources than it actually did and this way tries to charge a price higher than it should. Since it is

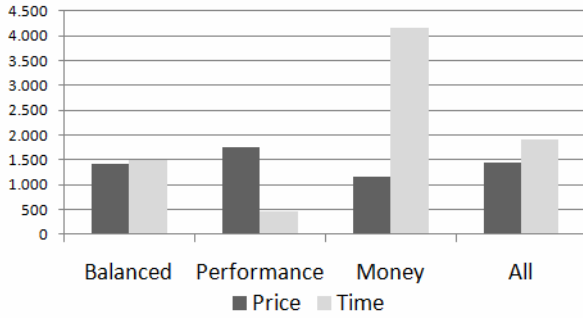
difficult to know how many resources an execution will consume without executing it, this situation is hard to detect. In order to detect it, it is assumed that the gridlets are passed by the consumer in groups with a relation among them, which is normal considering that a job will generate a set of similar gridlets [15]. Then, the price charged for the gridlets of the same job is compared in order to define an acceptance threshold and if one of the prices charged is higher than that threshold, it is considered over-priced. In this case the consumer also gives a bad classification and pays only the amount he considers fair. The bad classification is important because the reputation is inserted in the search. A producer with a lower reputation will have to lower the fee asked to compensate and eventually will stop being selected to execute the gridlets, which means that will not be able to gain credits to use the system.

On the other hand, the option of not paying gives the consumer the possibility of having some gridlets executed for free by falsely claiming that it was the victim of fraud. In extreme cases this would lead to the appearance of free-riders, nodes that never paid to use the system. In order to prevent this from happening, when the consumer submits a gridlet he has to pay a deposit, a value which is always paid to the producer that executed the gridlet. Moreover, after receiving the payment the producer also classifies the consumer. This is important because a consumer with a low reputation has to pay higher deposits. Also, a node is a consumer and producer, so a low reputation as a consumer will make him have a low reputation as a producer and vice versa.

## 4 Evaluation

In this section, we describe the evaluation of the model in simulation in order to determine its capability to match jobs to the available resources, how well the users' utility is fulfilled, and the behavior of the variable price definition policy. First, we describe the simulation environment.

For simulation, the model was implemented on the PeerSim simulator, using an event driven simulation. The simulation used 10 000 nodes with a Chord ID of 90 bits randomly generated. The predefined keys for the broker position were the multiples of 1025, which creates the presence of 123 brokers in the system. Table 1 presents the values of the characteristics used as the unit of cost. The characteristics of the nodes of the population were generated randomly. This generation had the following restriction: if a node had high values in one characteristic, it would have high values on all them. This is done because normally if a computer has a fast processor it also has a lot of memory space. In the beginning of all simulations, all the nodes start with 5 000 credits, so that they can pay for the jobs that they are assigned to submit into the system. This is done to avoid the long bootstrap time required if the credits were initial solely spread through transactions. The jobs consist of 10 gridlets all with the same size and the execution time is calculated as a function of the executor characteristics, which means that the higher the value of the characteristics the less time the execution will take. The time unit used is the tick, which is marked by PeerSim. The price is then calculated by applying the fee to the time the resources were busy.



**Fig. 4.** Results for the different type of users

**Job distribution:** First we tested the ability of the model to distribute the jobs over the available resources. For that, 40 000 gridlets are progressively inserted into the system, which is four times the system overall capability. The requirements of the gridlets have the preference for a node as less occupied as possible. As a result, when the system was full of gridlets it was achieved a node occupation of 100% and an average queue size of 3.9. Therefore we can conclude that the model is able to distribute the load evenly across all the nodes of the system.

**Consumer utility:** Next, in order to test the ability of the model to satisfy the users' utility, three types of users were created: balanced, performance and money. The balanced type of user wants to achieve a good balance between the execution time and the price paid. The performance type of user wants to get the execution done as fast as possible regardless of the cost. The money type wants to pay the least possible for the job execution. Figure 4 shows the average price paid and execution time for the three types of users. As it can be observed the type of users interested in having their jobs executed in the least amount of time possible, achieve an execution time three times faster than the time it takes to the balanced nodes. On the other hand the nodes which want to save the credits pay almost less twenty percent than the balanced nodes, however that comes with an execution time two and half times slower.

**Variable price policy:** The last aspect tested was how the variable price definition policy deals with variations of the demand and supply. Since we consider that the variations in the demand are cyclical, we present the results of the extent of two cycles (of 3000 ticks each) where the system was flooded with gridlets to be executed, during which the average fee asked was monitored. As it can be seen in Figure 5, the initial fee asked is of 50 and lowers a little because when simulation starts the system is empty, but after that the fee asked follows the trends of the occupation/utilization of the system. The average fee is raised while the system is heavily loaded and then lowered when it starts to have many nodes available, meaning that the offer is higher than the demand. This shows that resource management with our economic model is able to adapt to fluctuations in supply and demand. In greater demand, a premium is awarded to suppliers; with greater supply, an equivalent refund is awarded to consumers.

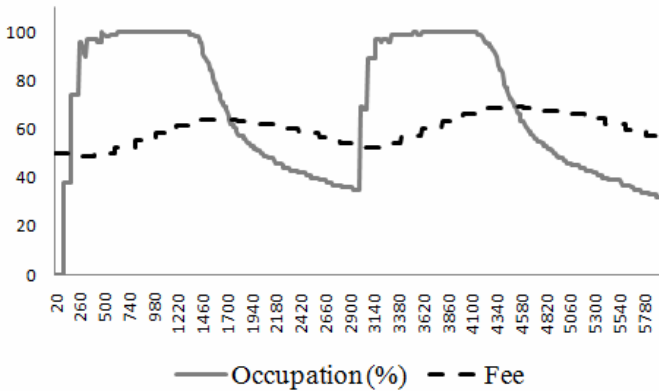


Fig. 5. Variation of the fee with the occupation

## 5 Conclusion

In conclusion the model presented provides a decentralized mechanism for the management of the resources in a cycle sharing peer-to-peer system which is able to distribute the jobs across the resources available in the system. In order to do that, it does not require any node to have global knowledge of the entire system. The resource management model allows each user to define its own requirements and is able to make the selection according to those requirements. This is an important aspect in a peer-to-peer system due to heterogeneity of users that can exist.

The model also offers incentives to resource contribution, since the only way of users gaining the credits they need to use the resources of the system, is by contributing themselves to the system. The incentives are improved by the variable price definition policy which creates a larger incentive for the contribution in times where the demand is bigger than the supply. Also, the use of the reputation system prevents misbehaving nodes of taking advantage of the system.

## Acknowledgement

This work was supported by FCT (INESC-ID multiannual funding) through the PID-DAC Program funds.

## References

- [1] Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys, CSUR (2004)
- [2] Buyya, R., Stockinger, H., Giddy, J., Abramson, D.: Economic models for management of resources in grid computing. In: Technical Track on Commercial Applications for High-Performance Computing, SPIE International Symposium on The Convergence of Information Technologies and Communications (ITCom 2001) (2001)



- [3] Buyya, R., Vazhkudai, S.: Compute power market: Towards a market oriented grid. In: The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001) (2001)
- [4] Carlo, L.G.L.M., Felipe, E.T.O., França, M.G.: The Use of Reciprocal Trade as a Model of Sharing Resources in P2P Networks. In: Proceedings of the 2009 Fifth International Conference on Networking and Services, vol. 00, pp. 91–96. IEEE Computer Society, Washington, DC, USA (2009)
- [5] Cheng, A., Friedman, E.: Sybilproof reputation mechanisms. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems, p. 132. ACM, New York (2005)
- [6] Dellarocas, C.: Analyzing the economic efficiency of eBay-like online reputation reporting mechanisms. In: Proceedings of the 3rd ACM Conference on Electronic Commerce, pp. 171–179. ACM, New York (2001)
- [7] Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th International Conference on World Wide Web, pp. 640–651. ACM, New York (2003)
- [8] Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, p. 53. Springer, Heidelberg (2002)
- [9] Paulino, J., Ferreira, P., Veiga, L.: Exploring Fault-tolerance and Reliability in a Peer-to-Peer Cycle-sharing Infrastructure. INFORUM (2010)
- [10] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, p. 172. ACM, New York (2001)
- [11] Rolli, D., Conrad, M., Neumann, D., Sorge, C.: An asynchronous and secure ascending peer-to-peer auction. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (2005)
- [12] Rowstron, A., Druschel, P.: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Liu, H. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
- [13] Silva, J.N., Ferreira, P., Veiga, L.: Service and resource discovery in cycle-sharing environments with a utility algebra. In: IEEE International Symposium Parallel & Distributed Processing (IPDPS 2010) (2010)
- [14] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, p. 160. ACM, New York (2001)
- [15] Veiga, L., Rodrigues, R., Ferreira, P.: GiGi: An Ocean of Gridlets on a “Grid-for-the-Masses”. In: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid. IEEE Computer Society, Los Alamitos (2007)
- [16] Vishnumurthy, V., Chandrakumar, S., Sirer, E.G.: Karma: A secure economic framework for peer-to-peer resource sharing. In: Workshop on Economics of Peer-to-Peer Systems. Citeseer (2003)

# Anatomy of Automatic Mobile Carbon Footprint Calculator

Ville Könönen<sup>1</sup>, Miikka Ermes<sup>1</sup>, Jussi Liikka<sup>1</sup>, Arttu Lämsä<sup>1</sup>,  
Timo Rantalainen<sup>2</sup>, Harri Paloheimo<sup>2</sup>, and Jani Mäntyjärvi<sup>1</sup>

<sup>1</sup> VTT Technical Research Centre of Finland

`firstname.lastname@vtt.fi`

<sup>2</sup> Nokia Research Centre

`firstname.lastname@nokia.com`

**Abstract.** A number of web-based systems provide environmental information to support our ecological lifestyle. However, there exists a need for more accurate and automated online systems facilitating personalised ecological awareness. This paper presents a travel type detection system for an automatic mobile carbon emission calculator. The system is able to detect automatically trips made by mobile users and provide them an easy and quick way for estimating the travel related CO<sub>2</sub> emissions. The system is based on energy and computationally effective multisource fusion in a mobile device. The paper introduces the system with design rationale, explains smartphone implementation and provides comprehensive evaluation from both accuracy and energy efficiency points of view. The system attained the overall travel type detection accuracy of **77%** and the daily average current consumption of **33mA**. The results show great promise for the developed methodology to facilitate 24/7 carbon emission estimation of a traveller.

## 1 Introduction

Modern people are increasingly environmentally conscious and green values affect their daily routines and behaviour. Currently, environmental sustainability information is available from various web-based systems. With these systems we can compare different options and adapt our lifestyle to a more ecological direction. Carbon dioxide (CO<sub>2</sub>) emission is among the most important environmental emissions that should be decreased. Carbon Footprint is a CO<sub>2</sub> measure of the effect of an individual on the Earth through her/his lifestyle. It consists of several main components: food, transportation, housing, general consumption of goods and waste. However, it is challenging for an individual to be aware on what is the effect of her/his daily selections on CO<sub>2</sub> emissions.

Travelling, including transportation, private and public travelling, is one of the main producer of CO<sub>2</sub> emissions. People carry their mobile phones with them all the time, and thus it is a motivating research challenge to develop an accurate mobile system for continuously detecting transportation types, for example car, bus, tram, train, etc., and estimate emissions according to the detected travel types. Such an approach brings particular research challenges, for

instance, just GPS may not be enough since the estimation of speed and location alone does not provide information for detecting means of transportation. For example, according to GPS-based system you may be moving on the street with certain speed but your travel type is unknown. It may potentially be a scooter, a motor bike, a bus, a car, a tram or even a bicycle. Those vehicles have very different CO<sub>2</sub> emissions. In order to detect particular travel types, more detailed characteristics on movements of a vehicle are required. Inertial sensors, accelerometers, gyroscopes and magnetic sensors may potentially provide hints on the movement characteristics. This brings in yet another research challenge, namely the critical energy consumption-issue in a mobile device since sensor signals require continuous monitoring and examination. Furthermore, a couple of research challenges are related to positioning. Firstly, usage of GPS, which is an inbuilt feature in modern smartphones, is extremely resource hungry. Secondly, positioning in urban environment is a challenge e.g. due to tunnels and high buildings. This may cause considerable errors to the GPS-based travel type detection system. The complementary solution for positioning is to intelligently fuse GPS with CellID-based method. In addition, to enable more accurate travel type detection, positioning should be fused with a real-time analysis of vehicle movements.

With these assumptions we can formulate research hypotheses for our empirical research approach for developing mobile travel type detection:

- By analyzing inertial sensors signals, mainly accelerometer signals, it is possible to find more detailed characteristics from the movements of a vehicle.
- By combining selectively GPS and CellID-based positioning methods, it is possible to provide error free and accurate enough positioning to support travel type estimation.
- By fusing positioning with the characteristics on movements of a vehicle it is possible to develop accurate travel type detection system.
- It is possible to develop energy efficient implementation of the travel type detection for a smartphone.

We start this paper by discussing existing literature and software in Section 2. In Section 3, we propose a structure for automatic travel type detection system and discuss its main components. In Section 4 we describe data collection for the evaluation of the proposed system and perform an evaluation. Finally, in Section 5, we conclude the paper. In the text, we refer to the set of methods developed for the trip and the travel type detection as Travel Type Detection Engine (TTDE).

## 2 Related Work

Already for several years, various carbon calculators have been available in the Internet. These calculators focus mostly on CO<sub>2</sub> emissions from travelling; the emission calculation from air travel being the most common, but often also other travel methods are supported. Often these carbon calculators are offered

by companies that also sell carbon offsetting services. At least 200 companies are selling carbon offsetting to private and corporate customers<sup>1</sup>.

The web-based calculators depend on user input, i.e., the end user must input the travels she or he has made. This kind of approach is problematic as the users do not remember or even know accurately, how much they have travelled with different travel methods during last weeks or months. Achieving accurate estimation of emissions requires inputting periodically your travel information. It would be tedious and it would require strong commitment from the user. Additionally, as some users would be more meticulous with their manual recording than others, this would render the data unreliable and comparison of the results pointless.

Also mobile phone based carbon calculators have emerged lately. These calculators typically use only GPS data to determine the travel method. The Carbon Diem and Ecorio can be considered to be the most advanced mobile carbon monitoring tools<sup>2</sup>. PEIR provides an alternative way to utilize location data. The mobile client sends location data to the server in which the trip and the travel type detections are performed. The system is discussed in [1]. However, the usage of location information for travel method detection leads to various problems: quite often, it is not possible to receive the GPS signal inside a vehicle, especially if the GPS is in an suboptimal location (e.g., in a pocket). Also, in some cases it is impossible to detect the travel method just by monitoring the geographical movement (a car and a bus driving the same route in a traffic jam).

GPS data related features are studied in [2] in which the authors propose several features for detecting travel type based solely on GPS data. The key idea in our paper is to find walking segments that are directly related to travel type changes. Suitable features for walking detection are studied in [3] and [4]. Using acceleration related features in activity segmentation is studied in [5]. Design guidelines of mobile applications promoting green practises are studied in [6].

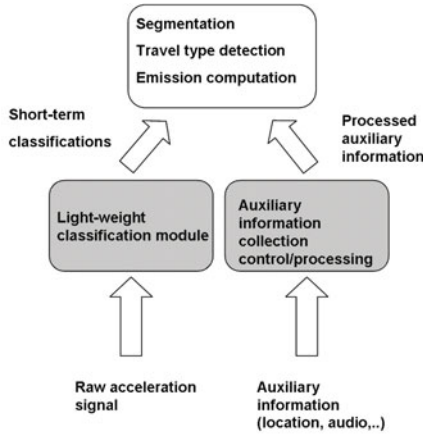
### 3 Overall Structure of Travel Type Detection Engine

Being able to detect relevant trips made by the mobile user requires extensive usage of the information sources available in the mobile devices. The detection system should be able to react rapidly to the changes in the environment, for example when the user starts to walk or leaves the office for driving home by a car. However, using the information sources such as GPS require a lot of battery power and hence relevant parts of the detection system should be carefully optimized. The key idea of the system presented in this paper is to find walking segments at first based only on acceleration data and then use the found walking segments to find segments containing travelling by motorized vehicles. This is possible because changing the travel type usually contains walking; people walk to their car or to a bus stop. A schematic view of the detection system is depicted in Fig. 1.

---

<sup>1</sup> <http://www.endscarbonoffsets.com/>

<sup>2</sup> <http://www.carbondiem.com> and <http://www.ecorio.org/>



**Fig. 1.** Overall structure of the travel type detection engine. Shaded regions in the picture represent modules that require extensive on-line computation

The primary component of the system is the *light-weight classification module* that performs short-term classifications based on the acceleration signal. This part of the system is running as a background process and the classification procedure is performed on-line. Hence the module is algorithmically light-weight and the implementation highly optimized. The other on-line component of the system is the *auxiliary information collection module* that periodically fetches location data from the cell network and GPS receiver. As the data fetching requires a lot of power, it should be done carefully by controlling the sampling frequency. Actual trip detection and travel type detection is done periodically by filtering short-term classifications and using auxiliary information. This component can be launched whenever new trips have to be visualized, for example upon user's request, periodically after some time period or after some event such as waking up from the power-saving state.

### 3.1 Light-Weight Classification Module

The input for the classification module is the 3D raw signal from the built-in acceleration sensor of the mobile phone. The acceleration signal is available only at the variable sampling rate (average 36Hz). At first the raw signal is buffered (buffer size approximately 3 seconds) and interpolated to the constant sampling rate, 50Hz in this study. The interpolated buffers are then categorized to four different categories (very high energy, high energy, low energy, very low energy) based on the signal energy that is computed from the total acceleration for enabling phone position invariance. The rationale behind the categorization is that human performed movements such as walking have higher energies than motorized movements such as train.

### 3.2 Auxiliary Data Collection

In addition to the acceleration data, also location data is recorded periodically in the side of the short-term classifications. The location information is then used to filter out misclassified travel segments due to random movements of the mobile phone and for getting more specific travel type detection. In addition, CO2 emission estimation is based on the travelled distance within each travel segment.

### 3.3 Segmentation and Travel Type Detection

The actual detection of the travel segments and the associated distance estimations occur in Segment Type Detection engine. In is not a real-time component and therefore related algorithms can be more complicated.

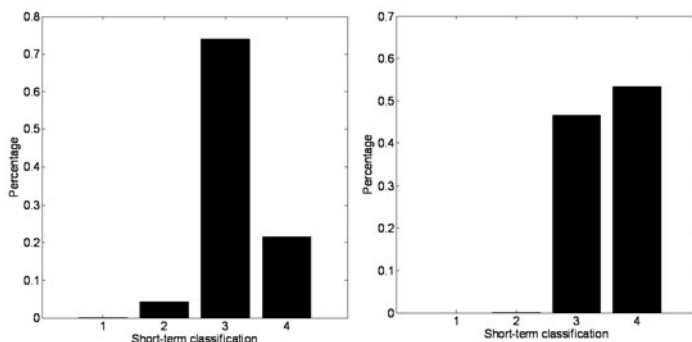
The input for the travel-type detection engine consists of the short-term classification and associated time stamps. The functionality of the engine can be summarized as follows:

1. Detect walk segments
2. The segments between walk segments represent potential travel segments
3. Compute a histogram from the short-term classifications for each non-walk segment
4. Compare the histogram to examples representing different travel types
5. Select the type with the best matching template
6. Isolate corresponding location marks for each non-walk segment
7. Compute several location based features from the location marks
8. Combine information from the location-based features with the selected class in (5) for determining the travel type

Different motorized travel types have their own characteristics. This is mainly due to their mass, for example rail vehicles have usually considerably higher mass than wheel based travel types. Also their acceleration profiles are different. For example metro accelerates very smoothly whereas accelerations of wheel-based traffic are more variable. From the data we have collected during the development of TTDE, the statistics shown in Fig. 2 are computed. The main difference is that the distribution is located more on the left with wheel-based movements and on the right with rail-based movements resulting from different masses of the vehicles.

When detecting actual travel types, we compare histograms computed from actual travel segments, i.e. segments between walking segments, with the templates shown in Fig. 2. The histogram comparison is based on Euclidean distance.

After the histogram matching phase, we have classified the segment to represent either wheel- or rail-based movement. The next phase is to determine a subtype inside these categories. If a segment is classified to represent wheel-based movement, we try to determine whether the travel type is private car or bus. In the case of rail-based movement, we try to detect whether the mobile user is travelling by tram, long-distance train or by some general, unclassified local



**Fig. 2.** Statistics representing distribution of different short-term classifications from wheel and rail based travel types. 1: Very high energy, 2: High energy, 3: Low energy, 4: Very low energy. Note that main difference is that the distribution is located more on the left with wheel-based movements and on the right with rail-based movements resulting from different masses of the vehicles.

rail-based travel type. The bus detection is done in the case there exists a clear waiting time period before the actual travelling. If the trip length is over one hour, the trip is classified to be long-distance travelling.

The following location-based features are computed for each travel segment:

- *Mean speed.* This feature is used for separating long-distance train from other types. The long-distance train has considerable higher mean speed than other travel modes.
- *Ratio between sudden changes and forward going parts of the segment.* This feature is used to filter out the segments that are not related to real travelling.
- *Linearity of the segment.* This feature is used to separate tram from other rail-based travel types. The linearity value of a tram segment is usually smaller than the value for other rail-based modes.

### 3.4 Distance Estimation and Emission Computing

As the major design goal of TTDE is that the travel detection process should be totally invisible to the mobile user, the phone may be positioned in many ways during travelling. For example, the phone may be located on the dashboard of a car or in the user’s pocket. The location may even change during travelling. Therefore GPS fixes are not always received and in some cases a travel segment may contain both GPS marks and network-based location information.

If no location marks are available, the distance estimation is only based on the duration of the trip and the estimated speed of the travel mode. In the case of network-based location information, the trip length is approximated by a linear segment between the start and end location. If real GPS marks are available, the trip length is computed as a sum of the linear segment lengths between the marks.

Current source of emission information for different travel types is LIPASTO-database<sup>3</sup> collected and maintained by VTT Technical Research Centre of Finland.

## 4 Test Settings and Performance Evaluation

We start the section by discussed the test settings in detail. Then we proceed to the detection accuracy of the system. We conclude the section by discussing the current consumption of the mobile application.

### 4.1 Test Equipment and Data Recording

For tuning the parameters of the travel type detection engine and validating the performance of the developed methods, two datasets were collected. The data collection software read the data from the mobile phone exactly in the same way as the real mobile application does, hence feeding the recorded data to the travel type detection engine, running either in the mobile phone or in the simulator, leads exactly to the same recognized trips as the on-line detection in the mobile phone. None of the data used for performance evaluation were used in the development of the system.

Acceleration and location data were collected with the embedded sensors in Nokia N95 8GB mobile phone. The test person used an annotation software running on an external PDA<sup>4</sup> to make realtime annotations about his current transportation method. The annotations were used as a reference information when evaluating the accuracy of the travel detection.

The positioning of the mobile phone during the tests was not fixed and its normal use was permitted during the tests. Most commonly the phone was kept in the pocket during the tests. The test dataset was collected by the one person only. However, we have noticed that the performance does not depend on the person carrying the mobile phone. Indeed, the system is more sensitive to the location of the device. For example, having mobile phone in the pocket of a heavy winter coat may hinder the detection of the walking segments by dampening the movements of the phone. These observations are based on our earlier experience on automatic user activity detection. Especially, detection of simple physical activities such as walking and running can be accomplished with high accuracy for the most users. This is important background also for this work as it is based on the walk segment detection.

Totally 12 recordings were made to evaluate the performance of the system. In a typical recording, the test person spent some time in the office, then left the office to travel to another destination, and returned to the office again. The total duration of the collected evaluation dataset was 28 h 52 min. The durations of the recordings varied between 1 h 1 min and 5 h 8 min.

<sup>3</sup> Available at <http://lipasto.vtt.fi/>

<sup>4</sup> The annotation system for PDA devices running Mobile Windows operating system is described in detail in [7].



## 4.2 Analyzation of Results

In Table [1](#) we inspect the performance of the system in discerning different travel types from each other. For each annotated travel type a distribution of the recognized travel types is shown. The number of missing detections is shown in the *not recognized* column of the table. The total number of annotations were 26 of which 24 were real travel segments and 2 were whole-day long recordings from office working and attending to a seminar. The system attained the total accuracy of **73%**.

**Table 1.** Confusion matrix of annotated and detected trips. LD stands for long-distance. Lines in the table represent the annotations and the columns actual recognitions. The total recognition accuracy is **73%**. Note that annotations tram, train, long-distance train and metro are recognized as general rail-based travel type, *rail*. In special cases, tram and long-distance train can be detected based on non-linearity of the travel segment and average speed, respectively. Annotation *no travel* contains two days data collected from office working and attending to a seminar.

Annotations	Detected travels								Total
	Car	Bus	LD bus	Tram	LD train	Rail	No travel	Not recognized	
Car	<b>0</b>	0	0	0	0	0	0	1	1
Bus	0	<b>5</b>	0	0	0	1	0	0	6
LD bus	0	0	<b>2</b>	0	0	0	0	0	2
Tram	0	1	0	<b>2</b>	0	<b>2</b>	0	0	5
Train	0	1	0	0	0	<b>1</b>	0	0	2
LD train	0	0	0	0	<b>2</b>	<b>0</b>	0	0	2
Metro	0	3	0	0	0	<b>3</b>	0	0	6
No travel	0	0	0	0	0	0	<b>2</b>	0	2

To obtain a deeper understanding on how well the start and end points of the detected travel segments correspond to the annotated trips, we align recognized trips with the annotations. Then we sum the start and end time differences between detected trip and the corresponding annotation. If both start and end times are shifted to the same direction, for example the trip is detected to start before it actually begins but its length is still correct, we subtract the differences from each other. As the emission values depend only on travelled distance that is directly related to trip length, the actual timing of an individual trip is not very significant. However, it directly affects user experience, the software should be able to detect trips and their timing accurately enough to convince user of the detection accuracy of the system. Formally, we evaluate the time shift by using the following metric:

$$\frac{|S_s \pm S_e|}{T},$$

in which  $S_s$  and  $S_e$  are the shifts in the start and the end times of a trip and  $T$  is the annotated length of the trip.

The second metric we use to evaluate the overall accuracy of the travel type detection is the percentage of the correctly aligned short time segments between annotations and detected trips. We split both the detected trips and annotations into

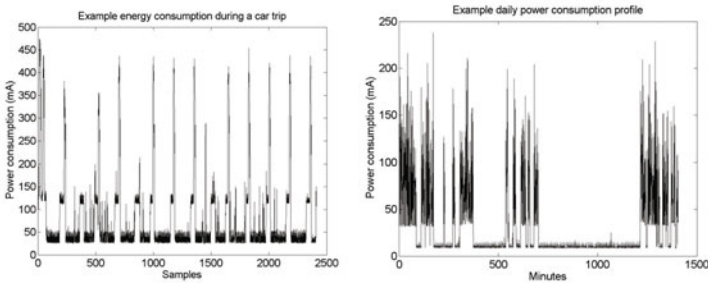
one second long time windows and then compute the percentage of the adjacent identically classified time windows in the whole set of classifications. Note that difference between this metric and the confusion matrix presented in Table 1 is that this metric also takes into account the length of the trips; misclassifying very short trip does not affect as much as misclassification of a long trip.

Over the whole dataset, the travel type detection system attained **77%** overall accuracy. The average timing error was **22%** when the mean trip length was 25 minutes.

### 4.3 Current Consumption

As the mobile application has several power saving features taking into account the actual state of the device and its user, current consumption depends heavily on the usage of the mobile phone. In Fig. 3 we present two example cases: a short recording from a short 30 minutes long car trip and a long full-day recording showing clearly how the power saving features affect to the average current consumption. All the measurement are done with the Nokia Energy Profiler tool<sup>5</sup> running on Nokia N95 8GB mobile phone which has 1200 mAh battery pack. In the short recording, we can easily recognize three current consumption states: (1) flat low current consumption areas in which the acceleration sensor is used as the only information source, (2) sudden increase in current consumption when trying to get a GPS-fix, (3) short, very high peaks related to network access for getting location information. As the network access times are very short, the major power consumer is GPS. When only the acceleration information is read and processed, the current consumption is around **40mA**. The average current consumption over the whole recording is **78mA**.

In the full-day recording, the power saving features can be easily seen. The large flat are in the image depicts night-time current consumption. Because the application read only acceleration sensor periodically and the time between periods is very long, the average current consumption is only **10mA**. The average current consumption over the whole recording is **33mA**.



**Fig. 3.** Two example cases depicting current consumption of the mobile application

<sup>5</sup> Freely available at <http://www.forum.nokia.com/>

## 5 Conclusions

The structure of the automatic Carbon Calculator software for mobile devices equipped with a built-in acceleration sensor and location information sources was presented in this paper. The implementation was then evaluated by using real-life data with detailed annotations. The system performed very well, the overall accuracy reached **77%**. The example implementation supports several power saving features and the current consumption of the system was tested with two power consumption scenarios, one measuring active state current consumption and one measuring current consumption over typical use scenario of the system. The active state current consumption was **78mA** and typical daily consumption **33mA**.

One approach to improve trip detection accuracy further is to add new information sources to the detection engine. Currently we are studying a possibility to use history data on previous trips made by a mobile user. If we are unsure about the travel type of a trip, we can try to predict it by using the information on travelling patterns of the user. Another approach we are currently investigating is to use a database containing information of bus, metro and tram stops to estimate travel types. Start and end locations of a trip are compared to the locations stored in the database and the type is then defined according to that information. The main challenge in this approach is the accuracy of the start and end locations of the trip. If this information is based only on the current CellID, the accuracy is too small for travel type detection. However, if the information originates from GPS, the approach works well and increases detection accuracy drastically.

## References

1. Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R., Boda, P.: PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In: Proceedings of The 7th Annual International Conference on Mobile Systems, pp. 55–68 (2009)
2. Zheng, Y., Quannan, L., Yukun, C., Xing, X., Wei-Ying, M.: Understanding mobility based on GPS data. In: Proceedings of the Tenth International Conference on Ubiquitous Computing (UbiComp 2008), pp. 312–321 (2008)
3. Ermes, M., Pärkkä, J., Mäntyjärvi, J., Korhonen, I.: Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *IEEE Transactions on Information Technology in Biomedicine* 12(1), 20–26 (2008)
4. Könönen, V., Mäntyjärvi, J., Similä, H., Pärkkä, J., Ermes, M.: Automatic feature selection for context recognition in mobile devices. *Pervasive and Mobile Computing* (2009) (in press)
5. Mäntyjärvi, J., Himberg, J., Korpipää, P., Mannila, H.: Extracting the context of a mobile device user. In: Proceedings of the International Symposium on Human-Machine Systems (HMS), pp. 445–450 (2001)
6. Froehlich, J., Dillahunt, T., Klasnja, P., Mankoff, J., Consolvo, S., Harrison, B., Landay, J.A.: Ubigreen: Investigating a mobile tool for tracking and supporting green transportation habits. In: Proceedings of 27th International Conference on Human Factors in Computing Systems (CHI 2009), pp. 1043–1052 (2009)
7. Pärkkä, J., Cluitmans, L., Korpipää, P.: Palantir context data collection, annotation and PSV file format. In: Proceedings of the Pervasive, Workshops, pp. 9–16 (2004)

# Empowering Elderly End-Users for Ambient Programming: The Tangible Way

Johan Criel, Marjan Geerts, Laurence Claeys, and Fahim Kawsar

Bell Laboratories, Alcatel-Lucent Bell N.V., Belgium  
{johan.criel,marjan.geerts,laurence.claeys,  
fahim.kawsar}@alcatel-lucent.com

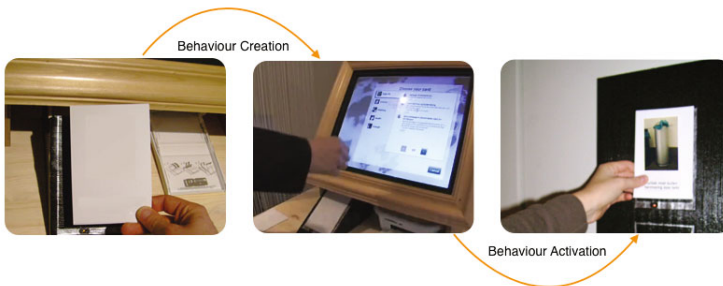
**Abstract.** This paper presents a novel tangible interaction technique leveraging Near Field Communication (NFC) enabled Magnetic Cards to empower elderly end-users in programming their pervasive computing environment. Following a user-centric design approach through cultural probes, we have explored the design space of possible programming opportunities for elderly users. Accordingly we have identified interaction techniques and candidate artefacts that could offer this user group a seamless programming experience for augmenting their ambient environment with software driven personalized behavior. Three aspects of programming - behavior generation, modification and (de) activation are addressed using two tangible user interfaces - i) NFC cards for smart behavior creation and modification and ii) a digital memo board as a placeholder for (de) activation of intelligent behavior. A qualitative feasibility study suggests that the proposed approach is simple, comprehensive and has the potential to be easily incorporated in the everyday routines of the elderly end-users.

## 1 Introduction

Pervasive computing has instigated a transformation of our environment into an interactive information sphere spread across time and space by combining sensing and software services with physical world. This essentially creates novel design opportunities for everyday artefacts - so called *smart objects*, as their behaviour are determined by software. This affords us to build imaginative new forms of interaction and functionalities that dramatically enhances their well-established features. While the technological building blocks for realizing such programmable physical space is converging there is a clear need to shift the focus onto end-users. It is essential to empower end-users with an appropriate capacity to program this interactive physical space populated with a range of smart objects and enable them to unleash computing enabled creativity capitalizing the possibilities offered by pervasive computing. Contemporary work has argued that it is very crucial to bring end-users into the center of control and management of pervasive computing environment [2, 13-15] as it is the people who occupy those environments and have the best knowledge about how their physical and computational environments should respond to their activities. Earlier research also demonstrated the prospect of involving end-users in creative engagement and its implication in raising end-users adoption and understanding of pervasive computing systems. For instance,

with the SpeakEasy platform end-users can search and compose semantically compatible appliances [7] and the Jigsaw Editor enables end-users to assemble jigsaw pieces representing pervasive services and smart artefacts to construct simple programs (e.g., connecting a doorbell to a camera for taking a photo shot when someone rings the bell) [1]. Also, a number of projects explored singular screen centric rule and recognition tools [4–6] to support end-users in personalising simple tasks, e.g., turning on/off lights contextually, etc.

However, unfortunately these solutions are not suitable for one class of citizen that represents a large population of our society : the seniors. In fact the diffusion of pervasive technologies in the home of senior citizens happens very slowly. Two primary reasons are: i) the technophobia of part of the elderly population and consequent anxiety towards technology usage and ii) the design principles of the application that fails to address elderly centric needs. [12]. The technology demand as well as interaction experience of senior citizens are fundamentally different than regular users, and an aspect like programming obviously puts significant burden on the designers. A classical evidence is that none of the projects mentioned above have addressed elderly-centric needs in their design. In this paper, we address this particular challenging issue of enabling elderly users to experience pervasive computing environment as well as program the environment to meet their customised needs.



**Fig. 1.** NFC-enabled magnetic cards are used to create customised behaviour from a range of templates and by setting appropriate conditions. This behaviour can be activated and deactivated by placing/removing the card onto/from a digital memo board.

Methodologically, we approach this in a top down fashion by taking a look at the life world of seniors and listening to the seniors themselves to have a profound understanding of what seniors want to do, how they want to do it in their ambient environment and with what granularity. This is unlike existing work in pervasive computing where tools were developed and given to end-users to configure their everyday lives in a pre-determined fashion without addressing their actual needs. This user centric approach gives us a deeper insight on elderly end-users requirement and affords us to design a solution that fits their need. By exploring the design space emerged from our insight, we have identified three aspects of programming that elderly users would like to be involved in : smart behaviour creation, modification and (de)activation. By technology probing, we converted these needs into a tangible interaction experience using simple cards and a memo board commonly found in our home. In particular, NFC-enabled magnetic cards allow elderly users to create personalized behaviour for their ambient

environment and a digital memo board acts as a place holder for active behaviour ensuring the metaphoric resemblance of a memo board and post cards (Fig 1). Examples of such custom generated behaviours of ambient environment are : *Setting reminder for grandma to put the garbage outside on Wednesday by switching on a lamp above the garbage can*; *Sending awareness SMS to caregiver when grandma opens the curtains in the morning, to relay that grandma woke up*, etc. In this paper, we present the design space analysis through cultural probes to understand such needs and possible opportunities for translating these needs into technology constructs. This is followed by the technical description of the tangible interfaces and interaction techniques emerged from our analysis. We also present a qualitative in-situ assessment of our proposed approach to highlight its strength and caveats.

The primary contribution of this paper is twofold - firstly, a design space reflection exposing the technology demands for creative engagement of elderly end-users and secondly a concrete design prototype of a tangible interfaces and corresponding interaction techniques to empower elderly end-users in programming the ambient environment.

## 2 Understanding the Design Space through Cultural Probes

End users are not programmers in the traditional sense. It is important to apprehend what end-users currently do, what they want to do as well as what technology can address those needs. A concrete understanding of these issues naturally leads to a technical solution that is more close to end user needs. In the context of senior citizens this is very crucial as their familiarity with technology as well as technical skills to interact with technology is relatively poor due to their age and possible age related diseases. Consequently we began our research by looking and listening to the seniors themselves. The contextual investigation we performed can be defined as an “applied ethnography” using two different methods: cultural probing and interviewing. 16 elderly persons who live alone (without partner) participated in the study - 8 senior persons were above 75 years old and the rest were above 60 years old. In addition, 8 caregivers for the participants with early dementia also participated in the study which allowed us to understand the technology demands of elderly that can be supported by their caregivers. All of the participants speak the Dutch language and live in the West-Flemish part of Belgium. The objective of their participation in the study was to formulate a qualitative understanding of the following three questions:

- What are the routines of senior citizens (with/without early dementia) who live alone?
- How do senior citizens who live alone adapt their in-home life?
- How do senior citizens with early dementia who live alone adapt their in-home life with the help of their caregivers?

The first phase of the study was carried out through cultural probing. The cultural probe that is used for ethnographic research, as in our case, is a kit that holds a variety of materials that trigger people to do individual research in their own lives [8, 9]. It is designed to engage participants in a study, and to stimulate them to response to the exercises that they find in the kit. The output of the participants is partly a source of inspiration to those involved in research and design i.e. ethnographers, designers and



**Fig. 2.** The Cultural Probe Kit (left) and the Interview Session (right)

engineers, and partly the departure point for a qualitative interview. The probe (Fig. 2) we used consists of a box full of nice items: a diary, a contact bulb, a helper-frame, a sheet to draw a home plan, stickers, a photo camera and the usual probe stuff - colored pens, a card with contact information, a notebook and some candy.

The diary forms the guideline of the probe. From day to day, the diary guided the participant through exercises on a certain domain. The domains were based on nine life fields including - day time spending, mental health, life style, life questions, physical health, education, social contacts, leisure time and residence. The elderly got three weeks time to use the probe kit before it was returned to the researchers. During that period, the researchers tried to stimulate the participant by sending a couple of postcards to his/her home address. On these post cards, additional questions were written, and each of them could be pasted inside the diary. The second phase of the study involved face-to-face interviewing with every participant. All interviews were coded (in total 994 codes were assigned) in the analysis software MAXQDA, analysed and discussed by the researchers and communicated to the rest of the teams.

## 2.1 Implications for Design

The investigation allowed us to gain a deeper understanding of the technology practices and demands of elderly end users. Considering the context of this paper we will limit the discussion in this section around end-user programming only. The detailed findings of the study is reported in other forums [10, 11]. In the following we report the implications of the study that lead us to formulate the design principles of our eventual solution.

**Plethora of Needs:** The elderly citizen, specially with dementia has a greater need for assistance even for rudimentary activities (e.g., brushing teeth, taking medicines, putting garbage outside, turning lamps on while reading, etc.) that seem obvious for other age groups. The three most prominent needs can be generalised as *situated assistance*, and support for *reminding* and *notifying caregiver*. The take away note is the fact that - it is imperative to empower seniors (and their caregivers) to generate customised solutions to address at least these needs.

**Ownership:** It was evident from our study that seniors want to have full control of their activities. They usually take extreme pride of little things they do, so-called “life hacks” to be in control of their lives and these tweaks also give them the feeling of ownership of the consequent effects in their lives.

**Object Centric Life:** Seniors in general structure their life by time. If they get dementia caregivers takes over the structuring of their life around some objects, so a change from time-centric life to object-centric life takes place. This essentially exposes the necessity for the instrumentation of everyday objects offering situated assistance.

**Intelligibility - Control and Consistent Feedback:** It is absolutely necessary to give full control to the seniors taking into consideration their capacity of cognitive and motor skills. Therefore, it is essential provide consistent, *large* and *visual* feedback that are recognisable to seniors so that they know what is happening around them and how they can control it. This is specially important for seniors suffering from dementia.

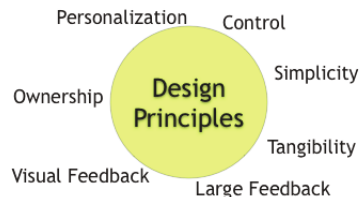
**Simplicity:** It is obvious that technology for seniors has to be straight-forward and should offer very basic functions that meet their needs. It is necessary to keep the balance between simplicity and rich functionality by carefully designing necessary features and keeping them relatively open to encourage endusers' active participation.

**Tangibility and Mechanical Experience:** It is essential to provide seniors with technology that offers a tangible and mechanical experience. Although current research is geared towards making tiny and invisible interfaces, for this class of user group it is still important to keep the tangibility and mechanical quality of experience. Pressing a button or turning a knob etc. is preferred over touch screen or other context driven triggers (e.g., position, location, etc.).

**Memo Board as the Control Panel:** Seniors organise their indoor and outdoor activities using a lot of memos and sticky notes pasted on the memo board or other flat surfaces (e.g., doors, refrigerator surface, etc.) as they need constant reminders of things to be done. Such practice gives us a justified choice to use memo board as the primary workplace where seniors can configure their technological assistance.

To summarise, the study clearly exposed the fact that - it is essential to empower elderly citizen with technology that allow them to create customised object centric solutions to address simple needs and get situated assistances in their lives. Accordingly, an imperative aspect of technology for ambient assisted living is involving elderly citizen closely with the technology for an intimate as well as vital interaction experience.

End user programming in its simplest form thus turns out to be a crucial component for ambient assistance. However, technology has to be simple, intelligible, personalized with large visual feedback and should give the ownership to seniors. Tangible interface has been identified as the major form of interaction due to its mechanical quality of experience. In addition, we have identified that the memo board commonly found in elderly homes is an ideal object that can be used as the workplace of seniors for creating their personalized technology solutions. In the next section, we discuss the technical translations of these principles into concrete technology prototypes.



**Fig. 3.** Design Principles to engage Elderly End-users in Creative Engagement



### 3 Tangible Interface and Programming Techniques

Following the implications of the study discussed in the previous section, it is evident that to involve elderly endusers in creative engagement we need to design the interface and corresponding interaction technique as simple as possible. We address this by identifying two common objects that can be manipulated in elderly home: sticky notes and memo boards, and augmenting them to enable simple tangible programming experience. The basic approach of our solution is to use NFC-enabled magnetic cards to create custom tailored behaviour and to place them on a digital memo board to activate or deactivate the behaviour resembling the metaphor of a sticky note pasted on the memo board.

#### 3.1 Tangible Interface

The user interface for supporting tangible programming is composed of three components designed following the principles mentioned earlier.

- NFC enabled Magnetic Cards: These cards are made of magnetic photo paper with a NFC chip attached on the back side of the paper (Fig. 4(a)). Once a behaviour is generated (discussed next) a *personalized* photo image sticker is printed that can be pasted on the front side of the card for *large, visual* and *recognisable* representation of the behaviour .
- Touch Screen Terminal for Behaviour Generation: A touch screen terminal is augmented with a NFC reader (Fig. 4(b)). This terminal provides support for creating and modifying custom tailored behaviour for ambient environment. In addition, a printer attached to this terminal allows printing *personalized* stickers that can be attached to the cards.
- Digital Memo Board: The digital memo board is equipped with five NFC readers on the back (Fig. 4(c)). The locations of these readers are displayed in the form of boxes on the front, to make it easier for the user to position the cards in the right way. The memo board is covered with a layer of small, round magnets to hold the cards. Below each box on the memo board, a Light Emitting Diode (LED) is placed to give *visual feedback*. This LED is illuminated when a card is placed on the board ensuring that the behaviour is correctly recognised and activated. The digital memo board can be mounted to the wall in the home on eye level of the user.

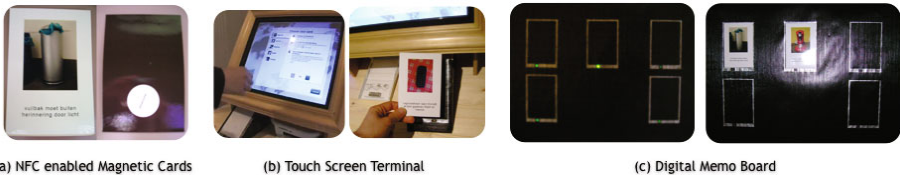


Fig. 4. Tangible Interface for Tangible Programming

An underlying infrastructure called CAEMP supports the tracking, interpretation as well as the consequent spontaneous federation of smart objects to facilitate these custom generated behaviour [16].

### 3.2 Tangible Programming Techniques

Three aspects of programming are addressed using the tangible interface presented above, these are - smart behaviour creation, smart behaviour modification and finally (de) activation of the behaviour.

**Tangible Interaction for Smart Behaviour Creation:** New smart behavior is created using the magnetic cards and the touch screen terminal (Fig. 5). The first step is placing a blank card on the touch screen terminal. Immediately, the smart behaviour creation environment starts up on the touch screen. The user can select a smart behavior template from the available template collection. An example of such a template is the *Show Media Template* that can be used to show assistive media on a screen if a person touches a specific smart object. The user can fill in the template with a selection of users, conditions, objects and actions. When the user is satisfied with the defined smart behavior, a description of the behaviour along with a custom defined representative image can be printed and can be attached to the card.

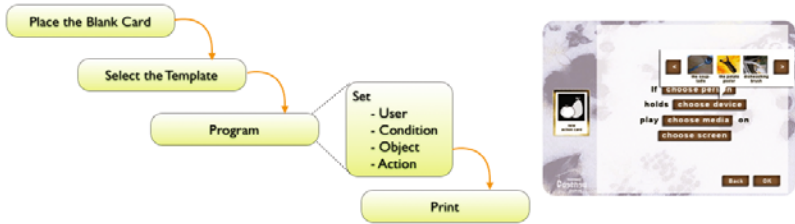


Fig. 5. Tangible Interaction for Smart Behaviour Creation

**Tangible Interaction for Smart Behaviour Modification:** Any existing smart behaviour can be modified or even deleted, using the touch screen terminal of the creation environment. To modify the smart behaviour, the representative card is placed on the terminal and immediately the programming environment is shown on the touch screen that end-users can manipulate (selection of users, conditions, objects and actions) to alter the old behaviour (Fig. 6). After confirmation, the old behaviour is replaced by the newly modified behaviour on the card.

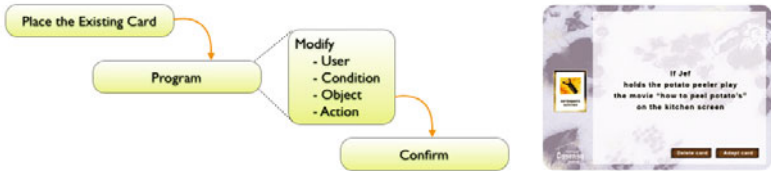


Fig. 6. Tangible Interaction for Smart Behaviour Modification

**Tangible Interaction for (De) Activation of the Smart Behavior:** To activate a new behaviour in the ambient environment, all users need to do is to place the card on the augmented digital memo board. Whenever a card is positioned in one of the frames on the memo board, the accompanying behaviour is active (Fig 7). The user can check if the card is positioned well, through the LED that is blinking under the frame. By simply removing the card from the memo board the user can deactivate its embedded smart behaviour.



**Fig. 7.** Tangible Interaction for Smart Behaviour Activation and Deactivation

As shown above, the combination of personalized magnetic cards and digital memo board interaction allows us to maintain the tangibility and simplicity of the programming experience. Additionally, particular emphasis is given to ensure that elderly endusers are put in the centre of control for the consequent behaviour as they can personalize, activate and deactivate these behaviours in a straight-forward fashion. Furthermore, large, visual and recognisable behaviour representation and consistent feedback through LEDs made sure that the interface and interaction experience meet the design principles. In the next section, we present an in-situ assessment of this tangible interface and corresponding programming techniques.

## 4 Feasibility Study in the Living Lab

The prototype tangible interface along with its underlying infrastructure [16] was installed in a living lab apartment that was furnished with smart objects, like a smart alarm clock, a smart toothbrush, a smart coffee machine, a smart garbage can, a digital photo frame in the living room and a digital screen in the kitchen. Both the touch screen terminal and the digital memo board were placed in that apartment<sup>1</sup>. A range of behaviour templates were available integrating these smart objects, for example : *Remind grandma to put the garbage outside on certain day by switching on a lamp above the garbage can, Start the television every day on a certain channel to remind grandma that her favourite television program is starting, Display pictures of grandmas wedding on the digital photo frame on every wedding anniversary*, etc. To qualitatively assess our prototype, 8 elderly participants (age range above 60 years) and 2 care givers

<sup>1</sup> Please have a look at <http://bit.ly/casensa> to experience our living lab, tangible interface and programming techniques as well as running use cases.

(all of them participated in the cultural probe session) were invited to visit the apartment. They were given basic information about the prototype and its objective, and were asked to perform certain tasks involving creation, modification and (de)activation of behaviours and then explain it to another participant. The test was done in a role-play format. At the end, the participants were interviewed and had to fill in a short survey. This study focused on the experience and usability of the proposed design. Our qualitative assessment of their performance and follow up interview clearly showed that all the elderly citizens were able to grasp the concept completely and found the process of programming straight forward. However, it was indicated that the behaviour creation phase could be relayed to the care givers, as sometimes elderly citizens got confused what can be done with the available objects. Once a behaviour is created, it was straight-forward for them to perform the other steps - modification, activation, etc. The large, visual feedback through picture and LEDs were found to be very crucial for their understanding. They also mentioned that these feedback gave them a sense of awareness and control of the environment. The major caveats of the system was the user interface of the touch screen terminal - the creation environment, as some screens' layout confused them due to the misplacement of buttons and corresponding pictures. In the later versions of the prototype we have addressed these issues.

## 5 Related Work and Conclusion

End User programming in pervasive computing environment is emerging as a hot area of research with the convergence of the technological building blocks of pervasive computing. Early seminal work on this area include Jigsaw Editor [1] that uses puzzle metaphor and allows non-expert users to configure pervasive services intuitively by assembling available components, and application of SpeakEasy computing platform that enables end-users to compose pervasive services spontaneously. Other work focused on the end user deployment activities and demonstrated that end-users understand the semantic association of physical devices and digital software [2]. Kawsar et al. presented a similar approach like ours using RFID Cards equipped with predefined custom behaviour. However, their work did not address the creation and modification aspects. In addition, we argue that our Digital Memo Board metaphor is more natural than their dedicated deployment tool [3]. Some work also addressed support for end-users personalization through rule or recognition based tools using graphical user interface (GUI). Rule based tools like iCAP [4], Alfred [6] provide visual tools, or sound macros to the end-users to define conditional rules based on the context to connect input and output events. Similarly recognition tools, or more formally Programming by Demonstration systems like CAPpella [5] use machine learning techniques to allow end-users to associate rules with real world events. These approaches are valid for rapid prototyping. However, we do not consider they are suitable for non-technical users, especially the elderly user group. Furthermore, these tools are designed for supporting very simple end-user tasks, such as turning on/off a device, loading presentation files, etc. These solutions primarily focused on facilitating the initial configuration tasks only, and it is not clear how well they can support custom tailored behaviour generation, modification and (de) activation as user requirements and environment change. In our proposed

approach, particular emphasis is given to a design approach that radically simplifies these activities though use of simple cards yet enabling the generation of personalized behaviour.

To conclude, we have presented a novel tangible programming interface and corresponding interaction technique specifically aimed at involving elderly citizens in creative engagement of programming ambient environment in a straight-forward fashion. Following a top-down user centric approach, we have presented an applied ethnographic study that helped us to formulate the design principles and corresponding technology prototypes. We also touched upon the qualitative assessment of our approach. We hope that, our technology prototype as well as the implications emerged from our multi phase studies will contribute in shaping the future of ambient assisted living technologies significantly.

**Acknowledgement.** The work reported in this paper was performed in the context of the ITEA2 AmIE project - funded by the Flemish IWT.

## References

1. Humble, J., Carbtree, A., Hemmings, T., Åkesson, K.-P., Koleva, B., Rodden, T., Hansson, P.: Playing with your bits: User composition of ubiquitous domestic environments. In: *UbiComp 2003*. LNCS, vol. 2864, pp. 256–263. Springer, Heidelberg (2003)
2. Beckmann, C., Consolvo, S., LaMarca, A.: Some assembly required: Supporting end-user sensor installation in domestic ubiquitous computing environments. In: Davies, N., Mynatt, E.D., Siio, I. (eds.) *UbiComp 2004*. LNCS, vol. 3205, pp. 107–124. Springer, Heidelberg (2004)
3. Kawsar, F., Fujinami, K., Nakajima, T.: Deploy Spontaneously: Supporting End-Users in Building and Enhancing a Smart Home. In: *The 10th International Conference on Ubiquitous Computing (UbiComp 2008)*, pp. 282–291 (2008)
4. Dey, A.K., Sohn, T., Streng, S., Kodama, J.: iCAP: Interactive prototyping of context-aware applications. In: Fishkin, K.P., Schiele, B., Nixon, P., Quigley, A. (eds.) *PERVASIVE 2006*. LNCS, vol. 3968, pp. 254–271. Springer, Heidelberg (2006)
5. Dey, A.K., Hamid, R., Beckmann, C., Li, I., Hsu, D.: A CAPpella: Programming by demonstration of context-aware applications. In: *ACM Conference on Human Factors in Computing Systems (CHI 2004)*, pp. 33–40 (2004)
6. Gajos, K., Fox, H., Shrobe, H.: End user empowerment in human centered pervasive computing. In: *International Conference on Pervasive Computing (Pervasive 2002)* (2002)
7. Edwards, W.K., Newman, M., Sedivy, J., Smith, T., Izadi, S.: Challenge: Recombinant computing and the speakeasy approach. In: *8th Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, pp. 279–286 (2002)
8. Graham, C., Rouncefield, M., Gibbs, M., Vetere, F., Cheverest, K.: How Probes Work. In: *Proceedings of the Conference of the Computer-Human Interaction Special Interest Group (CHISIG) of Australia on Computer-Human Interaction: Design, Activities, Artifacts and Environments, November 2007*. ACM Publisher, New York (2007)
9. Gaver, B., Dunne, T., Pacenti, E.: Design: Cultural Probes. *Interactions* 6(1) (1999)
10. Geerts, M., Criel, J., Claeys, L., Godon, M., Dieryckx, L., Deboutte, P.: Let the Seniors Hack! Finding daily life practices and hacks of elderly using cultural probes. In: *Workshop Capturing Ambient Assisted Living Needs, Ambient Intelligence Conference, Germany* (2008)
11. Geerts, M., Claeys, L., Criel, J., et al.: Seniors re-engineering context-aware applications. The casensa research process. In: *proceedings of AmI 2009 Conference, Salzburg* (2009)

12. Claeys, L., Criel, J.: Context aware computing: Future living as a social application. In: Withworth, B., Demoor, A. (eds.) *Handbook of Research on Socio-Technical Design*, pp. 779–793. IGI Global, London (2008)
13. Rodden, T., Benford, S.: The evolution of buildings and implications for the design of ubiquitous domestic environments. In: *ACM Conference on Human Factors in Computing Systems (CHI 2003)*, pp. 9–16 (2003)
14. O'Brien, J., Rodden, T., Hughes, M.R.J.: At home with the technology: an ethnographic study of a set-top-box trial. *ACM Transactions on Computer Human Interaction* 6(3), 282–308 (1999)
15. Edwards, W.K., Grinter, R.: At home with ubiquitous computing: Seven challenges. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) *UbiComp 2001*. LNCS, vol. 2201, p. 256. Springer, Heidelberg (2001)
16. Criel, J., Claeys, L., Trappeniers, L.: Deconstructing Casensa: The CAEMP Context-Aware Empowering Platform. *Bell Labs Technical Journal Issue* 16(1) (2011)

# Prototyping Augmented Traditional Games: Concept, Design and Case Studies

Tetsuo Yamabe, Takahiro Iwata,  
Takahiro Shichinohe, and Tatsuo Nakajima

Department of Computer Science and Engineering, Waseda University  
{yamabe,takahiroiwata,shichino777,tatsuo}@dcl.info.waseda.ac.jp

**Abstract.** Traditional game augmentation aims at adding new value and playful features to a traditional game with keeping its original look-and-feel. Players can concentrate on playing the game as usual, without paying extra attention to the service. Context monitoring enables to automatically record game events so that players can review a gaming process, strategy and excited scenes after game finishes, for example. Multimodal feedback also provides supporting information to understand the mechanism of games e.g., visualize threatened territories where beginner players tend to fail to notice. Moreover, players can dynamically and seamlessly turn on and off such pervasive gaming features while gaming. In this paper, we develop the concept of augmented traditional games with two case studies: Augmented Go and EmoPoker. We also report the preliminary user study results with implicating future work.

**Keywords:** Augmented traditional games, entertainment computing, tangible interaction, design guideline.

## 1 Introduction

While newly developed game consoles explore brand-new gaming style, traditional games, such as tabletop games, billiard and darts, are still appealing to us with various irreplaceable tastes. Spatial and physical interaction with tangible game objects enhances momentary, but emotional user experience in gaming. Players unconsciously use the all senses to perceive not only necessary information for the game play, but also the one that provides additional tastes or clues. For example, players need to read the face of a card to play poker. In addition, auxiliary information, such as shuffling sound, opponent players' facial expression and shakes caused by thrills, helps to play the game well and also simply makes game fun. Moreover, these advantages of traditional games cannot be reproduced in completely digitalized games, such as video poker.

On the other hand, the concept of mixed reality is getting realized today. For example, these days AR (augmented reality) toys are commercialized in the market. Users can interact with a virtual character by capturing a matrix code with a web camera. The character is superimposed on the code and animated as preliminary programmed. Real and virtual environments are united into a

hybrid world. Then digitally augmented features provide new user experience to consumers and assist players in gaming. For example, pervasive game is one domain in the mixed reality games [12]. The rapid progress of pervasive computing technologies enable games to be seamlessly integrated into our daily lives.

Such technologies often deserve attention because of the potential to invent brand-new games. However, they can also be used to augment existing games, and traditional games as well. In [3], Hinske *et al.* clarified a concept of augmented traditional game environments and pointed out the socializing aspects of the games. To play traditional games, usually players gather around a single table and directly share experiences in real-time. It smoothens social communication and amplifies the excitement caused by the game. Online games also enable users to simultaneously experience events, but the aforementioned additional information is lost through the indirect communication. Thus the augmented traditional game environments are basically designed with focusing on a single location gaming style; players are not supposed to move around, while most pervasive games adopt a time and location independent style [24].

Hinske *et al.* also proposed guidelines for the design and implementation of augmented traditional game environments. As they pointed out, it is challenging to augment an already existing game without changing its original look-and-feel. Moreover, in order to support a game flow with mixed reality technologies, we need to carefully design interaction: for example, context information should be observed passively so that players can concentrate on a gaming activity. Mixed reality technologies are expected to be applied to future products. Thus we believe that our findings can be also utilized in a wider range of applications as well as augmented traditional games. In this paper, we proceed with the discussion by prototyping two augmented traditional games. In Section 2, we explain the concept of our work with several use cases. Then in Section 3, we introduce the case studies of augmented traditional games: Augmented Go and EmoPoker. We also report preliminary user study results on Augmented Go system in Section 4. Lastly, in Section 5, we conclude the paper with implicating future work.

## 2 Augmented Traditional Game Environments

### 2.1 Attractiveness of Traditional Games

Traditional games offer physical interactions to game objects (e.g., darts, playing cards, Go stones). Physical interactions enable players to sense several kinds of information, even though they are not necessarily relate to the game itself. For example, one of important factors to evaluate other players' mental condition is pace. In digitalized games, a player's actions, such as selecting cards in poker and putting Go stones on a board, can be instantly performed as a single mouse click. There are certain distance between objects and players in physical games, thus it allows players to observe the duration spent for an action and get an idea of what the other player is thinking. Moreover, tactile interactions amplify expectation to uncertain things and the excitement caused by achieving a goal.



For example, in mahjong, players cannot know what kind of mahjong tile will be drawn in the next turn, since the tiles are ordered on the table as walls. During tile drawing motion (i.e., moving a tile from the wall to the player's area), the player's attention is drawn to the tile and expectation gets increased as the mark of the tile is gradually revealed. Skillful players can even recognize tiles by just feeling the surface of the tile. Therefore, the distance between the wall and the player's area can be considered to make the most exciting moment of the game play. These advantages of physical interaction cannot be reproduced in digitalized mahjong games.

We consider this most important factor of traditional games as *Ma*; Japanese word that represents distances among objects in a space. *Ma* is originally used in art and design, but it also can be used in a wider range of domains, such as martial arts and theatrical performances. In martial arts, players adjust distance to an opponent in order to keep their own territory to take good offense and defense. In theatrical performances, actors and actresses coordinate the timing of actions in order to make greater impression to audiences. *Ma* includes concepts of spatial distance, gap and pause, so both physical and mental distances among game objects including human players can be represented in this single word. At the same time, it is hard to reproduce *Ma* in digitalized games, since *Ma* is total information that players perceive in physical interactions.

## 2.2 Digital Augmentation with Pervasive Technologies

While digitalization causes the loss of the aforementioned advantages, pervasive computing technologies support various aspects of traditional games [2]. There are roughly two styles to apply the technologies to the domain of play and games: creation of novel forms of play and games, and augmenting existing forms of play and games. In [3], Hinske *et al.* mainly focused on supporting the players by providing services in digitally augmented traditional game environments, instead of integrating virtual play and game elements. They also proposed two important aspects of the augmentation: *the game flow virtualization of the game* and *the physical augmentation of the game*. The game flow virtualization enables an augmentation system to deal with a game rule so that it can check the rule consistencies and violations. Moreover, the system can provide information that is relevant to the game at appropriate timing. The physical augmentation also identifies two main objectives: unobtrusive technology integration and non-negative influence on the original game's rich social interactions. Table 1 quotes some of design guidelines they proposed in [3].

As the guideline indicates, ideally technologies support a gaming process with minimum interference. Moreover, technologies are supposed to be cognitively disappeared into background. As Hinske *et al.* pointed out, however, game objects are often small and will influence the choice of technologies. For example, RFID (radio frequency identification) tags are convenient to identify the objects. However, in order to detect the location of such small objects on a small tabletop, readers also need to be small or other positioning techniques are required. Flexibility is also a problem. If the tags are attached to cards, they should be enough

**Table 1.** Design guidelines for physical augmentation, partially excerpted particularly relevant items from [3]

1. The technological enhancement should have an added value.
3. The focus should remain on the game and the interaction itself, not on the technology.
4. Technology integration should be done in a way that is unobtrusive, if not completely invisible.
5. The game should still be playable (in the “traditional” way) even if technology is switched off or not working.
8. Players should receive simple and efficient access to information. Feedback should be immediate and continuous.
12. Secondary user interfaces should be minimized.

thin and bendable so that players can shuffle the cards without breaking them. Regarding to RFID tags, anti-collision mechanism is another issue. If a reader does not support it, the objects cannot be piled up and thus possible use cases will be limited. However, currently such anti-collision readers are still expensive. Thus physical augmentation could change game objects’ appearance, but players still should be able to play a game in the traditional way (even augmentation support is switched off).

Unobtrusive feedback and minimized (secondary) user interface are also important design issues to keep original look-and-feel. People often concentrate on game playing and experience a deep involvement that removes awareness of everyday life. According to [5], “*during play, distractions from major game tasks should be minimized by reducing nongame-related interactions and reducing the game interface to maximize the amount of screen taken up with game action*”. Interaction with an augmented game should follow this principle, in order to let a player experiences flow. However, it is difficult to automatically and perfectly recognize game relevant information. Thus players should be able to explicitly interact with the system to switch modes or correct detection errors for example. This user interface should be designed to naturally fit to original interaction process, rather than checking a display and configuring with a keyboard.

### 2.3 Use Cases

Augmentation enhances the value of traditional games, and it also allows using the game for the various purposes for which they are not originally designed. In order to proceed with the discussion with more concrete scenarios, we roughly categorized possible use cases into three domains: *Fun*, *Practice*, and *Research*.

**Fun.** One main purpose of traditional game augmentation is simply to enhance the pleasure of gaming and improve user experience. For example, Ishii *et al.* augmented a ping-pong table with a projector and sensor modules that detect a ping-pong ball’s position [6]. The system called PingPongPlus supports several modes and some of those transform game playing style from competition to collaboration. While such new gaming styles are invented with the augmentation, conventional ping-pong play is still enhanced with attractive visual effects such as virtual ripples on the table. Players do not need to newly learn how to use the

system. The system adds multimodal feedback corresponding to game events, and players realize they are interacting with the game (system) itself as well as opponent players. Multimodal feedback also can be used to make handicap between beginner and skillful players. For example, providing warning information prevents a beginner player from a careless mistake. We will explain more detail handicap making scenarios in Section 3.1 and Section 3.2.

One use case featured by context monitoring is automatic game event recording. The augmentation system automatically monitors context information such as game events, and players can review a process after finishing the game. Unless recording equipments (e.g., video camera) are preliminary set up, conventional gaming process is not recorded and cannot be replayed. However, players sometimes encounter a miraculous happening and it becomes unforgettable experience. We often regret that nobody has started filming, and actually we have missed lots of opportunity to look back on such memorable scenes. With the augmentation system, however, ordinary people can record their game play as a TV program relay a professional game match. Since we use original game objects, players can enjoy gaming as usual. If multimodal feedback is turned off. It is also interesting that the system automatically digests a recorded video from players' context information (e.g., a shout of joy), and proposes the "highlight of this game" after the game play.

**Practice.** While the fun domain mainly focuses on short-term game play, we consider the progress of gaming skill as another important factor to enrich longer-term user experience. As the concept of game flow indicates, a gaming process consists of multiple tasks and it has people concentrate on game play. A task has a clear goal, and the player would feel satisfaction by providing immediate feedback when the task is achieved. Game players can overcome difficult tasks and situations as master technical knowledge and skill. However, beginners tend to spend lots of time and effort for practice to advance their skill. They need a good trainer and iterative success experiences in order neither to be frustrated nor stop playing during a growth phase. Video and online games are one good way to practice game playing, but still it is difficult to check the point of strategy and understand the difference with skillful players. Moreover, as aforementioned, real atmosphere gives extra, but essential information such as facial expression to game players. We believe that real game experience with human players is important for a beginner player to learn game play.

Therefore, one possible scenario of the practice domain is the game review with skillful players. Automatic recording allows reviewing a finished game process with upper grade players. Particularly it is hard to record entire game events in imperfect information games, such as poker and mahjong, since a round finishes without disclosing most information to players. The recording system tracks the randomly dealt cards and moving tiles that are difficult to manually track, and then allows replaying the play with showing all players' hand. This "making invisible visible" approach is useful to grasp the game mechanism, as a player could get only decision results in a conventional way. Moreover, decision making under uncertainty elicits negative emotional arousal and increases cognitive bias.

Thus increasing the amount of information helps to notice own habit of decision making. It is also possible to present particular strategic patterns or information which novice players tend to miss, based on realtime game event tracking. Usually skillful players have learned useful intuition to insight the course of a game by experience. If such invisible experience could be visualized, a beginner player would recognize what expert players are seeing.

**Research.** Lastly, we emphasize that the augmentation system also contributes to academic research as well as individual game players. For example, Nacke *et al.* proposed to use a game monitoring system to analyze players' physiological responses [7]. Games have drawn considerable attention to understand human activity. Gaming process requires intelligent brain activities and also useful to know how people reacts to a game event and behaves under a certain situation. Moreover, while playing a game, people concentrates on a task and do limited interactions. It means that an observer can eliminate unnecessary factors in the experiment and expect tighter correlation between a game event and player's behavior than other types of wild experiments. Thus additional sensor devices, such as brain wave monitor, are also interesting to be supported by the system, even though they are not directly used to enhance gaming experience. In addition to the activity monitoring, it is also possible to make a stimulus and observe how people's behavior changes.

## 2.4 Related Work

In [8], Christian *et al.* introduced a computer-augmented card game. RFID tags embedded into cards are used to capture the game state; and players can check scores and advice with their mobile phone. Since card games are incomplete-information game, personal devices such as mobile phones are useful for giving each user individual feedback. However, as discussed in the paper, mobile phones also require visual attention and it leads to distraction from the game since players wanted to check system behavior and possible errors. Thus audio feedback might be useful to make lightweight feedback when simple notification should be given (e.g., confirmation of object tracking status).

In [9], Cooper *et al.* augmented one traditional game called Chinese Checkers to investigate user interface issues for tabletop augmented reality entertainment applications. Original game objects are completely replaced with digital devices and players manipulate virtual objects on a large display with an augmented reality marker. Even though their approach loses the advantages of traditional games we pointed out, unified interaction style for a wider range of applications is useful in some cases. Moreover, they introduced Passive detection framework, which is an object recognition infrastructure for pervasive services in a meeting room environment. Having a single setup for multiple games is important for decreasing the installation cost and increasing the availability of the framework.

## 3 Case Study

In this section, we introduce two case studies of the augmented traditional games: Augmented Go and EmoPoker.

### 3.1 Augmented Go

Go is a traditional board game for two players, where the objective is to occupy a larger portion of the board than the opponent. Black and white stones are used to control the territory and a board with a grid of 19 x 19 lines is used as the game field. The rules of Go are relatively simple, but the underlying strategies are extremely complex and rich. As in chess and reversi, numerous set sequences and strategies have been invented to reduce the complexity, but studying them requires the player to actually understand the strategic concepts. Thus it takes a long time for beginners to do well against experienced opponents. Augmented Go system supports several gaming modes in addition to normal play (Figure 1). The basic idea is to provide valuable information to beginners without additional interactions and devices. Figure 2 shows the system architecture of Augmented Go system. In this prototype, feedback is provided visually by superimposing guiding information onto the Go board with a projector. A web camera connected to a PC is used to detect the position of the Go stones. OpenCV library is used for visual analysis and the logic engine determines the information presented to the players about the current game situation. The sequence of stone moves is recorded into the database, which facilitates replaying the game for self-training. Finally, a visual animation is created by a flash application, and directly projected onto the Go board.

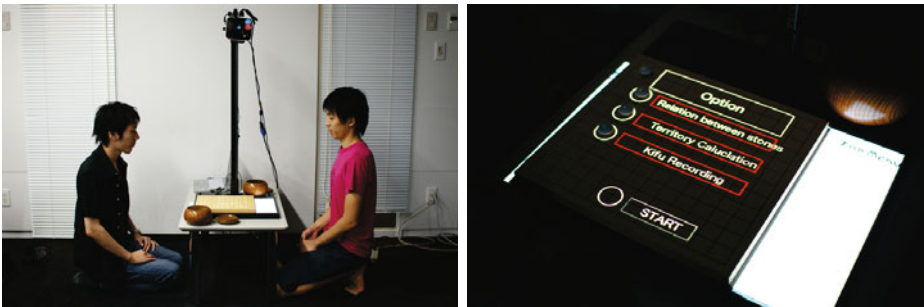
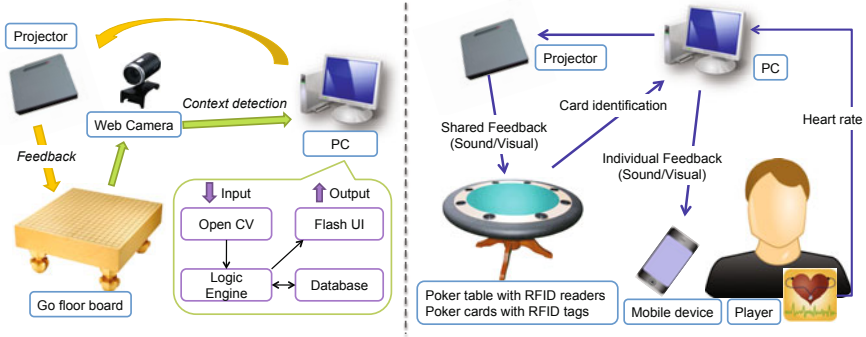


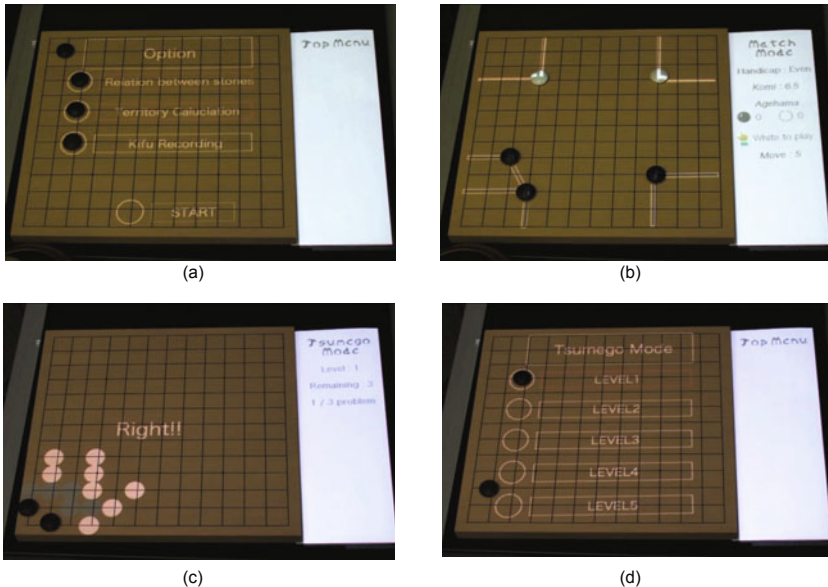
Fig. 1. Augmented Go system

As shown in Figure 3(a), players can interact with the system by putting Go stones on a menu that is projected onto a board. For example, *Match mode* represents the original concept of Go augmentation. In this mode, two players play Go as usual, but valuable information appears on the board to help beginners recognize the situation and make better decisions. The rules of Go are simple, but the vast number of possible moves in each turn makes it hard for beginners to make decisions. Moreover, on the large 19x19 board, beginners tend to concentrate on localized fighting and overlook the big picture in the process. It is difficult to recognize invaded areas, since an invasion process gradually progresses as new stones are put on the board. For choosing good offense and defense strategies, recognizing the links between the Go stones is important,



**Fig. 2.** System components of the augmented traditional games (left side: Augmented Go, right side: EmoPoker)

but it requires experience. Moreover, the match mode visualizes the strength of links between the Go stones. As shown in Figure 3(b), same-colored stones are connected with a line. Moreover, if a dangerous situation occurs somewhere on the board, a warning message appears to draw the player’s attention there to avoid losing the area. In addition, this mode supports stone position recording, so the players can review the match and discuss the efficiency of their strategy.



**Fig. 3.** Feedback examples of Augmented Go system

*Tsumego mode* is a type of exercise where the player is presented with a game situation, usually with the objective of finding the best sequence of moves in the given situation. In this mode, the positions of the stones are visualized on the board. Players can try out different moves by placing stones on the board, whereas the result and comments explaining key points are displayed as visual feedback (Figure 3(c)). Tsumego mode prepares different levels of questions, and a player can select in the menu (Figure 3(d)). As shown in these modes, the advantage of our approach is to allow players to get information through the original interaction offered by the Go board and the stones. By superimposing information onto the board, players can concentrate on the match at hand or self-training without fragmenting their attention towards an instructional book and etc. This is important to make it possible for the players to allocate enough cognitive resources for recognizing the situations in the game. Using original game objects as the basis preserves *Ma* and traditional look-and-feel, such as distance between players, touch of a wooden board and sound of stones.

### 3.2 EmoPoker

Poker is one of the most popular card games and it is often played for gambling. It maintains an element of chance because it is an imperfect information game: except for own hand cards, most of the cards are hidden from a player. A poker player needs to make decisions under uncertainty, such as changing cards, raising a bet and discarding a hand. Like other gambling games, this uncertainty can evoke the mixed feelings of excitement and anxiety. This makes poker play tightly linked with emotional arousal [10]. Beyond luck, as the common term “poker face” indicates, poker requires skill to control own emotional states. As learned from studies in behavioral psychology and behavior economics, emotional arousal increases the amount of irrational decision making [11]. This is a remarkable fact to consider when looking at gambling, since we also easily get emotionally excited in unpredictable economic activities. Thus even if a player prepares a well-formulated strategy, the strategy might fail as the consequence of the emotions elicited by the game.

EmoPoker system supports to objectively and calmly assess how aroused by providing biofeedback. Especially novice players may have a hard time paying enough attention to their emotions, as their cognitive resources are occupied by the demands of the game strategy. As shown in Figure 2, a wireless Bluetooth heart rate sensor is used to monitor the players physiological state. RFID readers and tags are used to track card moves on the table. The readers are embedded into the table, and thin tags are put on the cards face side (i.e., face-down cards are indistinguishable). Actuators are used to provide auditory and/or visual feedback to the poker players. For example, the heart beating sound is played by a mobile device. Such audio feedback could be shared among players, but in the self-training scenario, an individual player is assumed to be listening to the sound through earphones. It is also possible to provide individual visual feedback on the mobile phone (e.g., heart rate transition as an animation graph). A projector is used to display shared visual feedback to all players (Figure 4). In addition

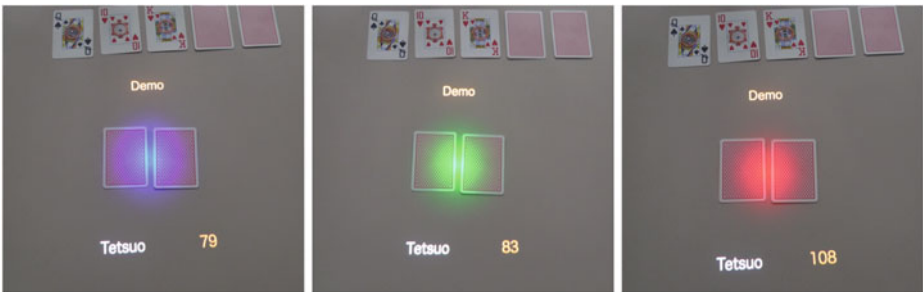


**Fig. 4.** EmoPoker system and visual feedback example. The projector spotlights the player’s hand cards and the color indicates arousal level (e.g., blue: calm, red: aroused). RFID tags are attached to each card in an unobtrusive way so that poker game events can be implicitly tracked and recorded by the system.

to audio feedback, arousal level can be conveyed visually. For example, when a player gets excited, EmoPoker spotlights them with ambient red light (Figure 5). In this case, players share the visual information, so this feature simply invents another poker playing style (i.e., system reveals poker face). This could also be a way to introduce a balancing factor between skillful players and novices. For example, if only the skillful players arousal levels are disclosed in the shared visual feedback, novices can use this hint to distinguish whether the experts are bluffing or not.

## 4 User Study

As a preliminary user study, the usability of augmented traditional game systems is evaluated. We also compared user experience between an augmented traditional game and a conventional digital PC game. We chose Augmented Go



**Fig. 5.** Feedback examples of EmoPoker system. As the user’s heartbeats quicken, the color of feedback changes from blue to green, and red. The feedback also blinks as a metaphor for heartbeat, and changes its speed according to the measured heart rate.



game for the study and prepared an extra PC game mode. The mode supports exactly same contents and functionalities, and the only difference is that physical interaction is replaced with digital one such as mice, keyboards, and displays. 12 subjects participated the user study and training tasks to learn the rudiments of Go game are assigned in each mode. The subjects are asked to answer a questionnaire after each task to subjectively rate user experience in 5-point Likert scale. We also collected comments and feedback from the subjects to identify future work towards more practical game augmentation.

For example, most subjects answered stone detection error was not obstructive to complete the tasks ( $Mean = 3.91$ , 5: strongly agree, 1: strongly disagree). As for the time lag in stone position recognition, it was also enough applicable ( $Mean = 3.58$ ). Visually projected information could be intuitively recognized and understood ( $Mean = 4.91$ ). The user interface of augmented Go game could be used as conventional PC games (i.e., there were no significant differences and difficulties from the extra PC mode) ( $Mean = 4.75$ ). On the other hand, the physical stone manipulation frustrated the subjects especially in repeated tasks, since they needed to physically remove stones to clear the board status and begin a new task ( $Mean = 4.00$ ). The most notable result in the questionnaire was that the physical interaction (i.e., putting stones onto the board) in augmentation mode promoted deeper elaboration rather than conventional mouse-click interaction ( $Mean = 4.25$ ). Some subjects answered that the transaction cost of redo and undo actions in physical interaction is relatively higher than digital interaction. Thus they stopped instant try-and-error learning approach, and started to choose the best hand with careful consideration. Moreover, the original game items such as Go stones and board developed the sense of real game, and feelings of actual match with human players.

As the limitation of this study, we could not confirm how much the AR training technique improves the performance of player's decisions. However, we could find that the augmented traditional game increased intrinsic motivation towards game rule learning. It would improve the efficiency of decision training as the motivation is crucial factor to acquire knowledge and develop skills particularly for self-learning processes. As a next step, we also need to explore how long the motivation continues by comparing with traditional training methods.

## 5 Conclusion

In this paper, we introduced the concept of augmented traditional games. In contrast to completely digitalized games, the augmented traditional games are developed with keeping traditional look-and-feel with original game items. We prototyped two case studies, Augmented Go and EmoPoker, as a proof of concept in order to clarify the possible use cases and advantages. We found that Augmented Go motivated subjects to study the game rules in the preliminary user study. In the future work, we will break down the experiments in order to investigate in the elementary design factors that differentiate user experience from conventional digital game play. We also develop a system framework to

generalize the system design and findings in the case studies for future traditional game augmentation.

## References

1. Hinske, S., Lampe, M., Magerkurth, C., Rocker, C.: Classifying pervasive games: On pervasive computing and mixed reality. *Concepts and technologies for Pervasive Games - A Reader for Pervasive Gaming Research 1* (2007)
2. Magerkurth, C., Cheok, A., Mandryk, R., Nilsen, T.: Pervasive games: bringing computer entertainment back to the real world. *Computers in Entertainment* 3(3) (2005)
3. Hinske, S., Langheinrich, M.: W41k: digitally augmenting traditional game environments. In: *TEI 2009: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (2009)
4. Jegers, K.: Pervasive game flow: understanding player enjoyment in pervasive gaming. *Computers in Entertainment* 5(1) (2007)
5. Johnson, D., Wiles, J.: Effective affective user interface design in games. *Ergonomics* 46(13&14), 1332–1345 (2003)
6. Ishii, H., Wisneski, C., Orbanes, J., Chun, B., Paradiso, J.: Pingpongplus: Design of an athletic-tangible interface for computer-supported cooperative play. In: *CHI 1999: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: The CHI is the Limit*, pp. 394–401 (1999)
7. Nacke, L., Lindley, C., Stellmach, S.: Log who's playing: Psychophysiological game analysis made easy through event logging. In: Markopoulos, P., de Ruyter, B., IJsselstein, W.A., Rowland, D. (eds.) *Fun and Games 2008*. LNCS, vol. 5294, pp. 150–157. Springer, Heidelberg (2008)
8. Floerkemeier, C., Mattern, F.: Smart playing cards – enhancing the gaming experience with rfid. In: *Proceedings of the Third International Workshop on Pervasive Gaming Applications - PerGames 2006 at PERVASIVE 2006*, pp. 27–36 (2006)
9. Cooper, N., Keatley, A., Dahlquist, M., Mann, S., Slay, H., Zucco, J., Smith, R., Thomas, B.: Augmented reality chinese checkers. In: *ACE 2004: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (2004)
10. Kallinen, K., Salminen, M., Ravaja, N., Yanev, K.: Psychophysiological responses to online poker game. In: *IADIS 2009: Proceedings of Game and Entertainment Technologies*, pp. 35–43 (2009)
11. Schwarz, N.: Emotion, cognition, and decision making. *Cognition & Emotion* 14(4), 433–440 (2000)

# Modeling and Experimental Validation of the Data Handover API\*

Soumeya Leila Hernane<sup>1,2</sup>, Jens Gustedt<sup>2</sup>, and Mohamed Benyettou<sup>1</sup>

<sup>1</sup> University of Science and Technology of Oran USTO, Algeria

<sup>2</sup> INRIA Nancy – Grand Est, France

**Abstract.** *Data Handover*, DHO, is a general purpose API for an efficient management for locking and mapping data. Through objects called *lock handles*, it enables to control resources in a distributed setting. Such handles ease the access to data for client code, by ensuring data consistency and efficiency at the same time. This paper explores DHO as it was presented in [1]. We model the phases that a lock handle crosses to achieve a DHO locking/mapping life cycle. The Grid Reality And Simulation (GRAS) environment of SimGrid is used as a support of an implementation of DHO and a series of tests and benchmarks of that implementation is presented. GRAS has the advantage of allowing the execution in either the simulator or on a real platform. For that purpose, we exploited a cluster of Grid’5000. The experiments that carried out cover various scenarios of sequences to lock a resources (inclusive or exclusive locking only, or combinations of both) and of combining different architectural factors. The tests demonstrate the ability of DHO to provide a robust and scalable framework. The good evaluation of the present work is consistent with an analysis of the expected behavior done by queuing theory.

## 1 Introduction and Overview

While large-scale distributed systems bring the advantages of an economy of scale, their heterogeneous structure limits the efficient use of system resources. Efficient data utilization in such systems relates to models and paradigms that emanate from two classes of architectures: shared and distributed memory. Both models handle data access and data consistency. However, they operate differently: threads are typically deployed on shared memory based programming environments (e.g OpenMP, [2]), while communication libraries based on the MPI API are widely used on distributed memory architectures [3]. Both paradigms have their powers and weaknesses.

The MPI-2.0 standard called Remote Memory Access (RMA) introduces One-Sided communication which gives the programmer the ability to write shared-memory programs in a distributed memory environment. Only one process is

---

\* Experiments presented in this paper were carried out using the Grid’5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see <https://www.grid5000.fr>).

involved in communication for both sides and each process exposes parts of its own memory for other processes. These parts of memory called "windows" specify regions of memory which have been made available for remote operations by other processes. By using one-sided communication, overhead is considerably reduced. However, it is not permitted use concurrent operations within overlapping windows. Also, consistency of conflicting accesses to the RMA is not guaranteed. This is why using this API is particularly tedious.

Our concept is most similar to POSIX' advisory file locking (`fcntl`) although these locks relate to a file descriptor rather than data [4]. `Fcntl` has some constraints. For example, locks associated with a file for a given process are removed when *any* file descriptor for that file is closed by that process. This occurs even if that file descriptor was never used to require a lock. Also, `fcntl` locks are not inherited by child processes and the order of lock acquisition is not specified. As a result, processes waiting to acquire a write lock may starve when waiting for successive read locks.

In this paper, we explore the Data handover (DHO) API that had been presented in [1]. Its goal is to be able operate efficiently in both paradigms as mentioned above.

DHO is designed to handle data on different types of systems such as multi-core machines, mainframes or in a distributed setting as clusters or grids. It enforces consistency of data and allows the application to handle individual parts (called ranges) of large objects. With this approach DHO extends known locking and mapping strategies.

DHO has not yet been implemented completely, the presented implementation is a first step towards this goal. [5, 6] introduced *ordered read-write-locks*, ORWL, for applications with iterative computations and parallel tasks. This model comes close to the locking strategy of DHO if we omit the capacity to lock parts of objects. A multi-threaded version of ORWL for shared memory were added to parXXL [7]. In [8], authors have extended classical distributed lock strategies to provide a byte-range locking in exclusive mode only.

In the following, the term resource is used for an instance of data, usually of large size that forbids an atomic access. The current implementation combines the acquisition of the lock of a resource with the provisioning of a copy of the data. The access to individual ranges of such a data object is not yet implemented.

The main objective of this paper is to measure the ability for an increasing number of requests to easily share a single resource in both *concurrent read* (cr) and *exclusive write* (ew) mode. The rest of this paper is organized as follows. Sec. 2 describes the core features of DHO. We provide an execution pattern by means of a deterministic finite state automaton (DFSA) in Sec. 3. Sec. 4 then presents the evaluation criteria and the experimental framework. Results are reported and discussed in Sec. 5.

## 2 The DHO Interface

DHO introduces an abstraction level between resource and memory through *lock handles* and according to an access control policy. Application processes (*clients*)

attempt to regain access to the resource locally. A resource owner (*server*) deals with requests, manages the resource and ensures its consistency. Clients may hold several handles to the same resource, e.g to regulate a sequence of accesses.

Through the handle, the clients request the instantiation of the resource in their local memory. This triggers events and crosses various states from request insertion until resource release. The state of knowledge about an acquisition (or not) that a client process has is denoted with capitalized names such as *Idle* for its initial state, see Fig. 1. States of the handle itself (that might be hidden to the client) are denoted in all minuscules like *invalid* for the initial state of the handle, see Fig. 2. The first action on a handle has always to be the `dho_create` function. The client attempts to open a socket to a server that holds the resource and waits for a reply. After that reply, the handle switches to the *valid* state.

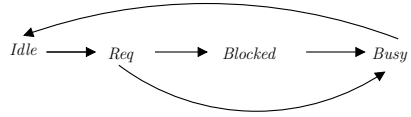


Fig. 1. States of a DHO client

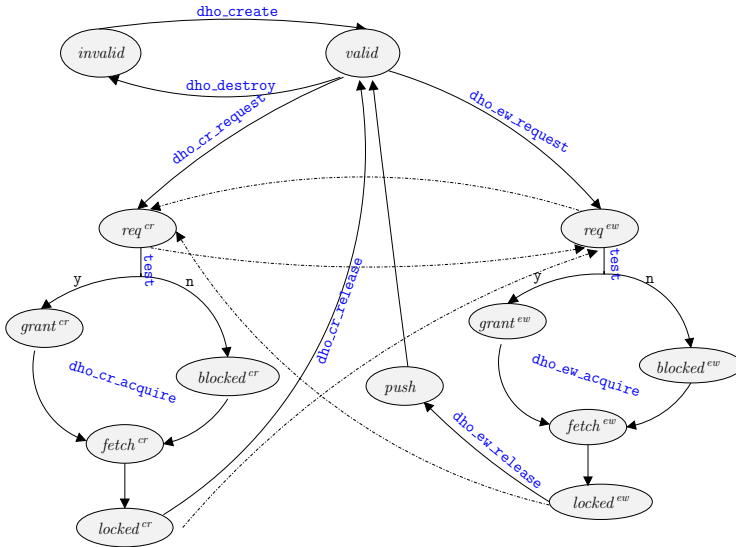


Fig. 2. State diagram of the DHO handle implementation

*The DHO life cycle.* The main phases of DHO and states through which a client and a handle go are designed to form a cycle (Fig. 2), the life cycle of a DHO locking and mapping request, a *dho cycle* for short. DHO distinguishes two access modes, namely a concurrent read mode (cr) for multiple simultaneous readers and an exclusive write mode (ew) for a single writer. A FIFO strategy assigns the resource to a handle in the same order as the requests are registered at the resource.

Phase 1 After an idle time  $T_{\text{idle}}$ , the client requests read access to the resource by calling the function `dho_cr_request` with argument `handle`. If a previous ew locking/mapping event is already present for the resource, the request is not serviced immediately, but is placed in a waiting queue. An ACK is returned by the function that confirms the registration of the request in the queue. The client switches to the *Req* state in non blocking mode and the handle becomes  $req^{\text{cr}}$ . A similar action is triggered by `dho_ew_request` for an ew access. In that case, the handle becomes thus  $req^{\text{ew}}$ .

Phase 2 When the resource is released from all previous read or write locking/mapping that inhibited the access in Phase [1](#) the handle switches to  $grant^{\text{cr}}$  respectively  $grant^{\text{ew}}$ .

By  $T_{\text{WaitGrant}}$ , we will denote the time to achieve this state. The client may at any moment ask to achieve the locking by calling `dho_cr_acquire` respectively `dho_ew_acquire`. If the lock is not yet obtained, the client is *Blocked* until this is the case. Then, the handle becomes  $blocked^{\text{cr}}$  or  $blocked^{\text{ew}}$ , respectively. The time that the client waits until that call will be denoted  $T_{\text{WBlocked}}$ . As an alternative to blocking, a call to the function `dho_test` can be used to know if access has been granted.

Phase 3 Once access has been granted, the handle switches to the states  $fetch^{\text{ew}}$  or  $fetch^{\text{cr}}$ , respectively. These are intermediate states during which the mapping is effectively done.  $T_{\text{fetch}}$  is the time required for this operation. In both cases, the handle then switches to the  $locked^{\text{cr}}$  respectively  $locked^{\text{ew}}$  state and instantiates the resource in local memory. The function returns a pointer to this local copy of the resource and the client becomes *Busy*. We will denote the times before a call to `dho_test` or that the client is *Blocked*  $T_{\text{WaitGrant}}$  and  $T_{\text{blocked}}$ .

Phase 4 After an application dependent lock time denoted by  $T_{\text{locked}}$ , the client calls `dho_cr_release` or `dho_ew_release`. If the request had been for writing, the eventual modification of the data is propagated to the resource during a transitory *push* state using a time  $T_{\text{push}}$ .

Then the resource is notified of the release of the lock and the handle becomes *valid*, again. On return from the call, the client switches again to the *Idle* state.

Thereafter, the client may again call `dho_cr_request` or `dho_ew_request` for the same handle. Listing [1.1](#) presents an overview of testing the DHO API.

At any time it isn't *Blocked*, the client can call `dho_test` to determine the state of the handle. The handle becomes *invalid*, again, by calling the function `dho_destroy` with a duration  $T_{\text{destroy}}$ . It is important that to note that  $T_{\text{WBlocked}}$  and  $T_{\text{locked}}$  are parameters imposed by the application and not observations.

*Other specific cases.* In addition to the above, DHO foresees all possible combinations of calls to its interfaces and acts accordingly. A client can call any function at any time and may not respect the logical order of the cycle as we have described above. The effect of calling the functions 'out of order' are that the handle

.5first is returned to the state of *valid* and that all priorities and locks are lost. Then the desired state is achieved as if the necessary other functions had been called previously. If for example the client has issued a read access that is not yet served or if it calls `dho_cr_request` while the process holds a lock and thereafter calls the function `dho_ew_request`, the lock or the old request is immediately canceled by the handle and the new request is inserted with a new ACK. The dotted lines in Fig. 2 illustrate such behavior.

**Listing 1.1.** An example of using the DHO API

```

char const* name;
dho_t *a;
double DELAY, T_WBlocked, T_Lock;
dho_create(name, &a);
do {
    dho_cr_request(a);
    sleep(T_WBlocked);
    dho_test(a);
    dho_cr_acquire(a);
    sleep(T_Lock);
    dho_cr_release(a);
} while(time < DELAY);
dho_destroy(a);
    
```

### 3 Formal Specification and Modeling

The resource can be only in three different states, namely that no request has been issued and that an EW or CR request has been granted. For the states of an DHO handle we already have implicitly introduced a model as Deterministic finite-state automaton (DFSA) [9]. Let  $M = (Q, \Sigma, \delta, q_0, F)$  be a 5-tuple, consisting of:

1. A finite set of states  $Q$  of a handle, where:

$$Q = \{ \textit{valid}, \textit{invalid}, \textit{blocked}^{\text{cr}}, \textit{fetch}^{\text{cr}}, \textit{grant}^{\text{cr}}, \textit{locked}^{\text{cr}}, \textit{req}^{\text{cr}}, \\ \textit{blocked}^{\text{ew}}, \textit{fetch}^{\text{ew}}, \textit{grant}^{\text{ew}}, \textit{locked}^{\text{ew}}, \textit{req}^{\text{ew}}, \textit{push} \}$$

2. A finite set of input symbols  $\Sigma$  corresponding to the DHO functions, where:

$$\Sigma = \{ \textit{create}, \textit{destroy}, \textit{test}, \textit{request}^{\text{cr}}, \textit{acquire}^{\text{cr}}, \textit{release}^{\text{cr}}, \\ \textit{request}^{\text{ew}}, \textit{acquire}^{\text{ew}}, \textit{release}^{\text{ew}} \}$$

3. A start state  $q_0 \in Q$  where  $q_0 = \textit{invalid}$
4. A set of accept states  $F \subseteq Q$ , where  $F = \{ \textit{invalid}, \textit{valid} \}$
5. A transition function  $\delta : Q \times \Sigma \mapsto Q_\delta$  is defined by the state transition Table II.

Let  $T_{\text{DHO}}$  denote the time of the whole DHO cycle. According to Fig. 2, it is easy to point out two possible paths for each access type, through which the handle goes. For example:

$$\{ \textit{req}^{\text{cr}} \rightarrow \textit{grant}^{\text{cr}} \rightarrow \textit{fetch}^{\text{cr}} \rightarrow \textit{locked}^{\text{cr}} \rightarrow \textit{valid} \}$$

This path occurs when the handle remains non blocking and often with the first requests or in such scenarios with many successive read accesses. The whole DHO cycle time is then:

$$T_{\text{DHO}} = T_{\text{WaitGrant}} + T_{\text{grant}} + T_{\text{fetch}} + T_{\text{locked}} + T_{\text{idle}}. \quad (1)$$

**Table 1.** State transition table. For readability, only the exclusive states are listed

	<i>valid</i>	<i>req<sup>ew</sup></i>	<i>grant<sup>ew</sup></i>	<i>blocked<sup>ew</sup></i>	<i>fetch<sup>ew</sup></i>	<i>locked<sup>ew</sup></i>	<i>push</i>	<i>invalid</i>
<b>create</b>	<i>valid</i>	<i>valid</i>	<i>valid</i>	–	–	<i>valid</i>	–	<i>valid</i>
<b>request<sup>ew</sup></b>	<i>req<sup>ew</sup></i>	<i>req<sup>ew</sup></i>	<i>req<sup>ew</sup></i>	–	–	<i>req<sup>ew</sup></i>	–	<i>invalid</i>
<b>test</b>	<i>valid</i>	<i>req<sup>ew</sup>   grant<sup>ew</sup></i>	<i>grant<sup>ew</sup></i>	–	–	<i>locked<sup>ew</sup></i>	–	<i>invalid</i>
<b>acquire<sup>ew</sup></b>	<i>valid</i>	<i>blocked<sup>ew</sup></i>	<i>fetch<sup>ew</sup></i>	–	–	<i>locked<sup>ew</sup></i>	–	<i>invalid</i>
<b>release<sup>ew</sup></b>	<i>valid</i>	<i>valid</i>	<i>valid</i>	–	–	<i>push</i>	–	<i>invalid</i>
<b>destroy</b>	<i>invalid</i>	<i>invalid</i>	<i>invalid</i>	–	–	<i>invalid</i>	–	<i>invalid</i>

When multiple requests are ahead of current write request, the handle crosses following states:

$$\{ req^{ew} \rightarrow blocked^{ew} \rightarrow fetch^{ew} \rightarrow lock^{ew} \rightarrow push \rightarrow valid \}$$

The corresponding DHO cycle time is:

$$T_{DHO} = T_{WBlocked} + T_{blocked} + T_{fetch} + T_{locked} + T_{push} + T_{idle}. \quad (2)$$

## 4 Experimental Parameters and the Simulation Environment

There are quite a lot parameters to be tuned before evaluating DHO. To facilitate the study, in all experiences we assume  $T_{idle}$  to be zero, or in other words, the client launches a new request as soon as the previous lock is released. An M/M/1 queuing model formalizes DHO’s mechanism, namely a single-server queue model corresponding to a queued (FIFO) access to regulate the access to a resource, see [10]. In our context, two kind of requests are inserted in the queue. We take  $N_c$  clients as sources for arrivals of requests. Let  $T_{Wait}$  denote the waiting time of a request in queue, *i.e.* the time between the call to request and the return from fetching into state *locked*:

$$T_{Wait} = T_{WaitGrant} | T_{WBlocked} + T_{grant} | T_{blocked} + T_{fetch}. \quad (3)$$

Let  $T_{Blocking}$  denote the time the client is blocking before acquiring the resource, *i.e.* the time between the call to acquire and the return from the fetching into state *locked*:

$$T_{Blocking} = T_{blocked} + T_{fetch}. \quad (4)$$

Experiments to measure these waiting times were run under the *Grid Reality And Simulation environment*, GRAS, [11]. GRAS is socket based and is a powerful API that gives the opportunity to implement distributed applications on



top of real platforms. Indeed, with GRAS, we can deploy experiments on real platforms without even modifying or recompiling the code. We just have to relink the program. Under simulation mode, we exploited a description of a realistic platform, namely *gdx* which was a subset of Grid’5000. The simulation uses two nodes of that virtual cluster for its own purpose, for the server process and another to compute the different delays. The remaining 118 nodes had a role as clients ( $N_c = 118$ ). Here is the description format:

```
<cluster id="gdx" prefix="gdx-" suffix=".orsay.grid5000.fr"
radical="1-120" power="3.185E9" bw="1.25E8" lat="1.0E-4"
bw_bb="1.25E9" lat_bb="1.0E-4"/>
```

As usual, latencies accumulate by their sum, whereas the bandwidth of a chain of links is their minimum. We use the improved SimGRID model [12] for communication to guarantee a good accuracy for messages greater than 100 KiB:

$$T = \alpha \cdot L + \frac{S}{\min(\beta \cdot bw, \gamma')}, \quad (5)$$

where  $\alpha = 10.4$  and  $\beta = 0.92$  are two correction factors.  $\gamma'$  sets the window size ( $\gamma_{\text{TCP}}$ ) in TCP connections,  $\gamma' = \gamma_{\text{TCP}}/2L$ .  $L$  is the accumulated latency ( $L = 2\ell + \ell_{bb}$ ). So, each flow really starts after  $\alpha L$ . The TCP flows achieve 92% of the physical bandwidth. We use this improved model, since we are testing different sizes. We set the  $\gamma_{\text{TCP}} = 10^7$ .

## 5 Simulation Results and Analysis

Dho was run successfully as well in reality as in simulation. However, this section relates to a series of benchmark we have conducted in the simulation framework and is summarized as follows:

We evaluate fetching and updating phases and the DHO cycle time. We focus at first, upon write locks only by making variation of the lock time and other delays (see Sec. 4). Thereafter, we look for the impact of the other application delay  $T_{\text{WaitBlocked}}$ . We plan then, scenarios of read accesses and of both. An analysis is done finally through the queuing system. As described in Sec. 2, the creation is the first phase that allows the handle to negotiate a *valid* state. Conversely, the destruction phase allows the handle to return to the *invalid* state. We study the influence of the latency on the times  $T_{\text{create}}$  and  $T_{\text{destroy}}$ . We took three values both for  $\ell$  and  $\ell_{bb}$ :  $10^{-5}$ ,  $5 \cdot 10^{-5}$  and  $10^{-4}$ (s).  $T_{\text{create}}$  is at least  $145.6 \cdot 10^{-5}$  and at most  $145.6 \cdot 10^{-4}$ . We also made similar variations to measure the destruction phase. The delays are between  $72.8 \cdot 10^{-5}$  and  $72.8 \cdot 10^{-4}$ . We observed the following relationship:

$$T_{\text{create}} = \alpha \cdot (2L + \ell + 2\ell_{bb}) \quad (6)$$

$$T_{\text{destroy}} = \alpha \cdot (L + \ell). \quad (7)$$

$\alpha$  is the parameter as above in the network model. These results are consistent since destruction involves less communication between client and server.

Next, we study the time during which the mapping/locking  $T_{\text{fetch}}$  and the updating  $T_{\text{push}}$  is done. We varied latency and bandwidth as previously as well as the resource size from 100 KiB to 1 GiB. We took the lock time  $T_{\text{locked}}$  in the range  $0 \dots 10$ . Experiments were performed with clients that run 100 cycles.

Despite all variations, we observed that the push delay is exactly the communication time  $T$  as explained earlier in Sec. 5. Also, the fetch time is close to the push time:

$$T_{\text{fetch}} = T_{\text{push}} + \alpha \cdot L. \quad (8)$$

These results clearly demonstrate the quality of the model since all the bandwidth is consumed.

We aim at evaluating the DHO cycle time and to measure its adequacy in extreme circumstances. A first series of experiments were done with ew requests only. To better analyze the behavior of the API, we first had  $\overline{T}_{\text{WBlocked}} = 0$  such that the handle switches to the blocking state as soon as the request is issued. The clients launched 100 write accesses and 100 executions were performed. In Table 2 we observe first that the waiting time is roughly equal to the blocking time. This is consistent with the formulas in Sec. 4. Secondly, we observe that the gap between durations for small resources (100 KiB and 10 MiB) is minimal. This is explained by the fact that, the mapping ( $T_{\text{fetch}}$ ) and update ( $T_{\text{push}}$ ) times are negligible (formulas 3 and 4). Also, durations are longer when the resource size and the lock

**Table 2.** Times (s) for  $\text{BW} = 1.25 \cdot 10^8 \text{ B/s}$ ,  $L = 10^{-5} \text{ s}$

Resource size	$\overline{T}_{\text{locked}}$	$\overline{T}_{\text{Wait}}$	$\overline{T}_{\text{Blocking}}$	$\overline{T}_{\text{DHO}}$
100 KiB	0-10	$5.833 \cdot 10^2$	$5.833 \cdot 10^2$	$5.883 \cdot 10^2$
	10-20	$17.449 \cdot 10^2$	$17.4491 \cdot 10^2$	$17.599 \cdot 10^2$
	20-30	$29.164 \cdot 10^2$	$29.164 \cdot 10^2$	$29.414 \cdot 10^2$
10 MiB	0-10	$6.024 \cdot 10^2$	$6.024 \cdot 10^2$	$6.075 \cdot 10^2$
	10-20	$17.67 \cdot 10^2$	$17.67 \cdot 10^2$	$17.821 \cdot 10^2$
	20-30	$29.336 \cdot 10^2$	$29.336 \cdot 10^2$	$29.587 \cdot 10^2$
100 MiB	0-10	$7.942 \cdot 10^2$	$7.942 \cdot 10^2$	$8.002 \cdot 10^2$
	10-20	$19.583 \cdot 10^2$	$19.583 \cdot 10^2$	$19.742 \cdot 10^2$
	20-30	$31.282 \cdot 10^2$	$31.282 \cdot 10^2$	$31.541 \cdot 10^2$
1 GiB	0-10	$27.569 \cdot 10^2$	$27.569 \cdot 10^2$	$27.712 \cdot 10^2$
	10-20	$39.20 \cdot 10^2$	$39.444 \cdot 10^2$	$39.444 \cdot 10^2$
	20-30	$50.86 \cdot 10^2$	$50.86 \cdot 10^2$	$51.204 \cdot 10^2$

time are larger. The difference of delays is not significant for the big sizes. Whereas the cycle delay is  $31.541 \cdot 10^2$  for the resource of 100 MiB for example, it is  $51.204 \cdot 10^2$  for a size 10 times larger. The lock time dominates the DHO cycle in these cases. The queue length ( $\overline{q}$ ) is about 116 in all cases. Thus, the DHO cycle time depends on the lock times  $T_{\text{locked}}$ ,  $T_{\text{fetch}}$ ,  $T_{\text{push}}$  and the queue length:

$$\overline{T}_{\text{DHO}} = [T_{\text{locked}} + T_{\text{fetch}} + T_{\text{push}} + \alpha \cdot \delta(3\ell + \ell_{bb})](\overline{q} + 1) \quad (9)$$

where  $\delta \simeq 7$ . The service time noted  $1/\mu_1$  can be deduced from 9:

$$\frac{1}{\mu_1} = T_{\text{locked}} + T_{\text{fetch}} + T_{\text{push}} \quad (10)$$

From formula 9, we can infer the mean time of the cycle for solely cr accesses:

$$\overline{T_{\text{DHO}}} = [T_{\text{fetch}} + \alpha \cdot \delta(3\ell + \ell_{bb})](\bar{q} + 1) + T_{\text{locked}}. \quad (11)$$

Are counted neither lock time of previous requests nor update time  $T_{\text{push}}$  in such a cr mode. Similar executions on the basis of read accesses only, were conducted and confirmed. The service time ( $1/\mu_2$ ) is approximately  $T_{\text{fetch}}$ . Results obtained so far have demonstrated the scalability of the DHO model, regardless of simulation time, resource size and the large number of requests to handle.

Let us recall that up to now, we assumed that  $T_{\text{WBlocked}} = 0$  in all our experiences. In asynchronous computations,  $T_{\text{WBlocked}}$  is the delay the client takes before launching the DHO acquire function. We now study the influence of this parameter with set of experiences that fix a  $T_{\text{locked}}$  delay and a resource size of 100 MiB. As shown in Fig. 3, the time remains almost the same, despite the different delay parameters we chose. In fact,

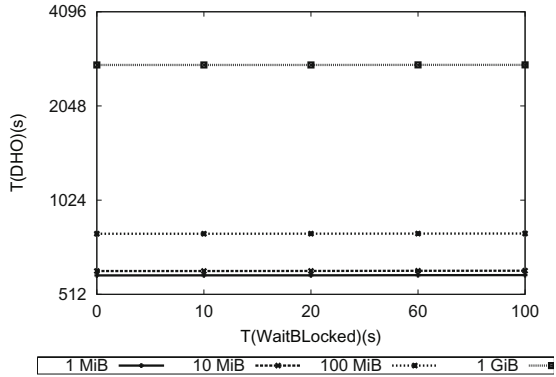


Fig. 3. DHO cycle vs  $T_{\text{WBlocked}}$ ,  $T_{\text{locked}} = 5$ ,  $size = 100\text{MiB}$

the gap between  $T_{\text{Wait}}$  wait and  $T_{\text{Blocking}}$  extends. Thereby, if  $T_{\text{Wait}}$  increases, the blocking time decreases and vice versa. This is interesting because no additional overhead is measured in the cycle. Actually, applications that use the API expect a delay before switching to blocking state.

The last scenario relates to a random sequence of both types of requests. We took  $T_{\text{locked}}$  and  $T_{\text{WBlocked}}$  within the range  $0 \dots 10$  s. For a size of 100 MiB, the measurements reported as the cycle delay, the waiting and the blocking times are respectively 446.35, 441,024 and 412.92. The queue length is approximately 114. Since our system is modeled as an M/M/1 queuing system, we analyze the occupancy of the model ( $\rho$ ). It is defined as the average arrival rate ( $\lambda$ ) divided by the average service rate ( $\mu$ ) such that,  $\mu = \frac{\mu_1 + \mu_2}{2}$ . With Little's law [13] and formulas 11 and 9  $\rho$  is  $\frac{(\bar{q}+1)\mu}{T_{\text{DHO}}}$ .  $\rho$  reached  $\simeq 0,4$  which means that the system has reached a stable state.

## 6 Conclusion

This paper presents a study of the DHO model according to different aspects. The life cycle of DHO has been detailed by a elaborated modeling. From measurements of different scenarios in which the two kind of accesses were issued,

we conclude that DHO is a stable system. What is interesting, is that our model allows to issue asynchronous locks and then provides overlap between computation and control. Also, the queuing formalism has allowed us to confirm good performance of the current implementation.

In spite of successful results, more studies are needed in the future. Soon, we plan to compare simulated results with those obtained within Grid'5000. In ongoing work we investigate the locking of ranges of the resource by different handles. Also, we are looking for a formal modeling and a validation of a distributed model so as to improve DHO mechanism in terms of the DHO cycle time.

## References

- [1] Gustedt, J.: Data handover: Reconciling message passing and shared memory. In: Fiadeiro, J.L., Montanari, U., Wirsing, M. (eds.) Foundations of Global Computing, Dagstuhl, Germany. Dagstuhl Seminar Proceedings, (05081) (2006), <http://drops.dagstuhl.de/opus/volltexte/2006/297>
- [2] The OpenMP API specification for parallel programming, <http://www.openmp.org>
- [3] Arquet, P.: Introduction à MPI – Message Passing Interface (2001), <http://www2.lifl.fr/west/courses/cshp/mpl.pdf>
- [4] Fcntl, <http://pubs.opengroup.org/onlinepubs/009695399/functions/fcntl.html>
- [5] Clauss, P.-N., Gustedt, J.: Iterative computations with ordered read-write locks. *Journal of Parallel and Distributed Computing* 70(5), 496–504 (2010)
- [6] Clauss, P.-N., Gustedt, J.: Experimenting iterative computations with ordered read-write locks. In: Danelutto, M., Gross, T., Bourgeois, J. (eds.) 18th Euromicro International Conference on Parallel, Distributed and network-based Processing, pp. 155–162. IEEE, Italy Pisa (2010)
- [7] Gustedt, J., Vialle, S., De Vivo, A.: The parXXL environment: Scalable fine grained development for large coarse grained platforms. In: Kågström, B., Elmroth, E., Dongarra, J., Waśniewski, J. (eds.) PARA 2006. LNCS, vol. 4699, pp. 1094–1104. Springer, Heidelberg (2007), <http://hal-supelec.archives-ouvertes.fr/hal-00280094/en/>
- [8] Quinson, M., Vernier, F.: Byte-range asynchronous locking in distributed settings. In: 17th Euromicro International Conference on Parallel, Distributed and network-based Processing - PDP 2009, Weimar, Germany (2009), <http://hal.inria.fr/inria-00338189/en/>
- [9] Sipser, M.: Introduction to the Theory of Computation. PWS, Boston (1997)
- [10] Smith, W., Taylor, V., Foster, I.: Using run-time predictions to estimate queue wait times and improve scheduler performance. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1999, IPPS-WS 1999, and SPDP-WS 1999. LNCS, vol. 1659, pp. 202–219. Springer, Heidelberg (1999)
- [11] Casanova, H., Legrand, A., Quinson, M.: SimGrid: a generic framework for large-scale distributed experiments. In: IEEE International Conference on Computer Modeling and Simulation - EUROSIM / UKSIM 2008. IEEE, Cambridge (2008), <http://hal.inria.fr/inria-00260697/en/>
- [12] Velho, P., Legrand, A.: Accuracy study and improvement of network simulation in the SimGrid framework. In: Simutools 2009, pp. 1–10. ICST, Brussels (2009)
- [13] Little, J.D.: A proof for the queuing formula:  $L = \lambda W$ . *Operations Research* 3(9), 383–387 (1961)

# Formal Modelling and Initial Validation of the Chelonia Distributed Storage System

Sami Taktak and Lars M. Kristensen

Department of Computer Engineering  
Bergen University College, Norway  
{stak@hib.no,lmkr}@hib.no

**Abstract.** A storage system with file replication is an important element in supporting reliable and fault tolerant file access in many grid computing systems. The Chelonia distributed storage system is being developed in the context of the NorduGrid project. It provides transparent access to replicated files stored on a heterogeneous collection of storage nodes with files being organised in a global name space. Our contribution is to develop a formal specification of the operations supported by the Chelonia system using the Coloured Petri Nets modelling language with the aim of verifying functional correctness. An important contribution of our formal modelling approach is to abstract from the concrete data stored on the storage nodes within the system. This caters for verification of the storage operations using finite-state model checking techniques.

## 1 Introduction

Grid computing systems constitute prominent examples of distributed systems exhibiting a very rich behaviour due to the synchronisation and communication between a large set of concurrently executing components. The design and implementation of distributed software providing advanced grid services is therefore a challenging task, and identification of design or implementation errors causing a grid system to malfunction can be a very challenging task. Considering the critical services provided by grid systems for problem solving in business, science, and engineering motivates the development and application of techniques for building reliable grid computing software. This in turn motivates the use of formal techniques and tools for obtaining precise specifications of grid systems and supporting protocols, and for verifying their functional correctness prior to implementation and deployment. Recently, there has indeed been an increased interest in the formal modelling and verification of grid systems [2, 5, 11, 13, 15].

Coloured Petri Nets (CP-nets or CPNs) [12] is a formal method for the specification, simulation, and verification of distributed systems. It is a graphically-oriented modelling language capable of expressing concurrency, communication, non-determinism, and system concepts at different levels of abstraction. CPNs are a combination of Petri Nets and a programming language: Petri Nets [19] are used to model concurrency, synchronisation, and resource sharing; Standard ML [20] is used for modelling data manipulation and for creating compact and

parametrisable models. Formal modelling with CPNs are supported by CPN Tools [4] which provides support for construction, simulation, functional and performance analysis of CPN models. CPNs have been successfully applied for formal modelling and verification in a wide range of application domains [10].

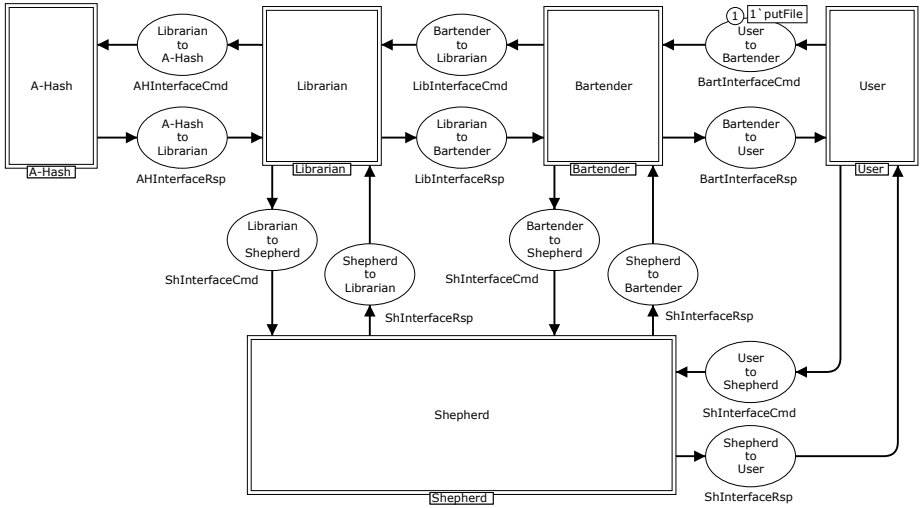
The Chelonia distributed storage system [17] is being developed in the context of the NorduGrid project and supports a replicated file service where files and collections (directories) are stored on a set of storage nodes. The Chelonia system provides commands for, e.g., listing the content of collections, and for downloading and uploading files. The Chelonia system is based upon four main services (components): a *Bartender* service providing the high-level interface for clients (users) of the system, a *Librarian* service managing the storage system hierarchy of files, a *Shepherd* service for providing access to back-end storage elements, and an *A-Hash* service storing meta-data about files and collections.

Recently there has been increased interest in modelling and verifying services used in grid computing [1, 7, 9, 14], but few have focused on verification of storage resource management services. A conceptual model of the Storage Resource Manager (SRM) Interface Specification [8] has been presented in [6]. The model in [6] describes the interface from the user point of view and has been used to verify interoperability between different implementations of the SRM specification. In this paper, we present a model of the *internal behaviour* of a storage resource manager for the Chelonia system, and show how one can model such a complex system using CPN. This model represents the behaviour of internal services and their interactions in the Chelonia system while abstracting from the actual data stored within the system. Chelonia was chosen since it is a system currently under development and it has a well documented internal structure [18]. The main challenge in modelling Chelonia for model checking was to avoid the state space explosion problem. Hence, we have focused on abstracting the model from actual data while preserving the behaviour of the whole system.

The paper is organised as follows. Section 2 gives a brief introduction to the Chelonia system by presenting the top-level modules of the constructed CPN model. Section 3 presents selected parts of the CPN model, and Section 4 shows how the operations provided by the Chelonia system have been formally verified using model checking techniques. Finally, in Section 5 we provide conclusions and discuss future work. We assume no prior knowledge of CPNs nor the Chelonia system. This paper informally introduces CPNs using the constructed CPN Chelonia model as an example. Readers interested in a complete introduction to and formal definition of CPNs are referred to [12]. Due to space limitations, we present only representative parts of the constructed CPN model [16] in this paper.

## 2 The Chelonia System

The CPN model of the Chelonia system consists of 16 modules organised into 4 hierarchical levels. Figure 1 shows the top-level module of the CPN model representing the most abstract view of the system provided by the model. The double



**Fig. 1.** The top-level module of the CPN Chelonia system model

outlined rectangles in Fig. 1 are *substitution transitions* representing compound behaviour. The CPN Chelonia model has one substitution transition for each of the services of the storage system: the Bartender, the Librarian, the Shepherd, and the A-Hash services. In addition to these, the User substitution transition models the users invoking the commands provided by the storage system. Each substitution transition has an associated *submodule* where the detailed behaviour of the component is specified. The name of a submodule associated with a substitution transition is written in the small rectangle positioned below each substitution transition. Here, an associated submodule always has the same name as the substitution transition (but this is not generally required).

The substitution transitions in Fig. 1 are connected via *places* which by convention are drawn as ellipses. We use places to model interaction point enabling communication between the components in the Chelonia system. Interaction between two services are modelled by two places: one place modelling the sending of a request to the service and one place modelling the reception of a response from the service. Each place has an associated *colour set* (a data type) determining the kind of *tokens* that can reside on the place. As an example, the place *User to Bartender* (top right) has the colour set `BartInterfaceCmd`. Figure 2 provides the definition of selected colour sets used in Fig. 1. The colour set `BartInterfaceCmd` is an enumeration type with a value for each of the operations that can be invoked on the bartender service. The colour set `BartInterfaceRsp` is a union colour set representing the responses that can be received from that service.

The *current state* (also called a *marking*) of a CPN model consists of a distribution of tokens on the places of the CPN model. In this case, there is a single token with the value `putFile` on the place *User to Bartender* representing the user invoking the put-file operation for adding a file to the system. All other

---

```

colset BartInterfaceCmd = with
  stat | getFile | delFile | putFile | unlink | move | modify |
  listCollection | makeCollection      | unmakeCollection ;

colset BartInterfaceRsp = union
  statRsp      : BartResponse + getFileRsp : BartResponse +
  delFileRsp   : BartResponse + putFileRsp : BartResponse +
  unlinkRsp    : BartResponse + moveRsp    : BartResponse +
  makeCollectionRsp : BartResponse + unmakeCollectionRsp : BartResponse +
  listCollectionRsp : BartResponse + modifyRsp      : BartResponse;

colset BartResponse = with completed | failed;

```

---

**Fig. 2.** Selected colour set definitions for the CPN module in Fig. 1

places in the depicted state are empty, i.e., contain no tokens. At the top-level of the CPN Chelonia model, the components as represented by the substitution transitions and the associated submodules exchange information by producing (and consuming) tokens on (from) the connected places. More generally, the execution of a CPN model consists of *occurrences* of transitions consuming tokens from input places and adding tokens to output places. The values of the tokens consumed from input places and produced on output places are determined by the *arc expressions* on the arcs connecting transitions and places.

An advantage of the graphical modelling approach provided by CPNs is that the communication architecture between the components in the system become explicit at the top-level of the model which at the same time captures the overall structure of the Chelonia system and the subservices that it relies on.

### 3 Example: The Bartender Service

This section presents our modelling approach by showing how representative parts of the Bartender have been modelled. In particular, we introduce the key abstractions made in order for the constructed models to become independent of the data stored in the storage elements of the system.

The Bartender is responsible for interfacing user requests with the components of the Chelonia systems. As represented by the `BartInterfaceCmd` data type (Fig. 2) users are, e.g., able to request file information (`stat`), to list collections (`listCollection`), to put new files (`putFile`) into the system, and to retrieve files (`getFile`). Figure 3 depicts the Bartender submodule associated with the Bartender substitution transition in Fig. 1. The places `FromUser` (top), `ToUser` (bottom), `FromLibrarian`, and `ToLibrarian` which have an `In` or an `Out` tag next to them are *port places* and represent the interfaces of the Bartender. The port places are linked to the *socket places* connected to the Bartender substitution transition in Fig. 1. As an example, the port place `FromUser` is linked to the socket place `UsertoBartender` in Fig. 1 which implies that the two places will



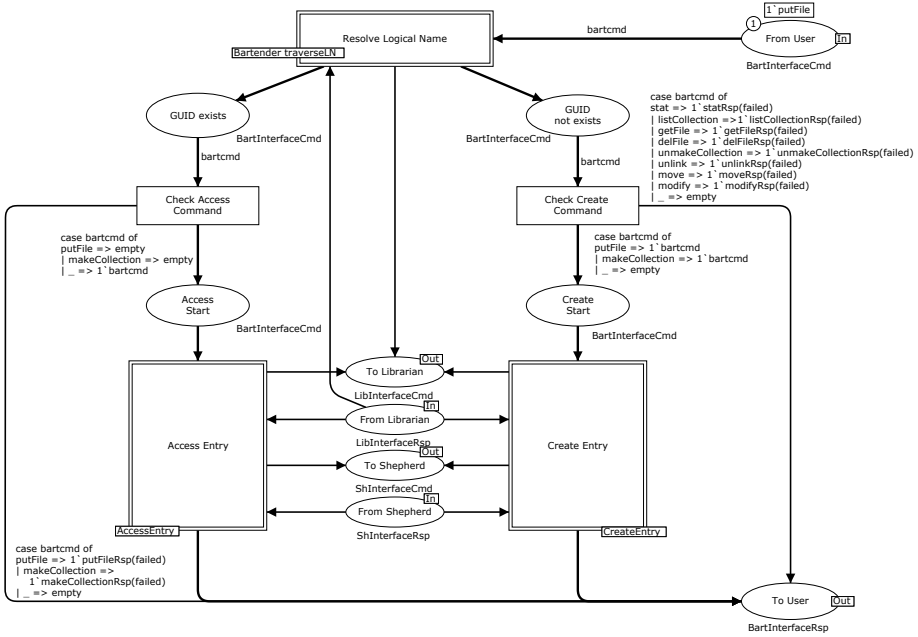


Fig. 3. The Bartender module

always have identical markings. In this case, the `putFile` token present at the `UserToBartender` socket place in Fig. 1 is also present on the associated port place `FromUser` in Fig. 3. This means that a module and its environment interact by adding and removing tokens on port and socket places. The other port places in Fig. 3 are linked to socket places in Fig. 1 in a similar manner.

When the Bartender receives a command, the first step is to *resolve* the *logical name* (LN) provided by the user into a *globally unique identifier* (GUID). This mapping from LNs to GUIDs is modelled via the submodule of the substitution transition *ResolveLogicalName*. In Chelonia, all meta-data associated with files and collections are stored in a database and accessed through the A-Hash interface. Each entry in the database (typically a file or a collection) is referred to by its GUID. But, from the user point of view, each entry is accessed through its LN which is similar to file paths known from, e.g., UNIX. A LN is a bounded string giving the absolute path of an entry. As example, the LN of a file `myfile` belonging to the collection `user` which resides in the root collection `/` will be `/user/myfile`. Internal commands refer to entries in the database through their GUID. The Librarian provides an interface between the A-Hash and the other services and includes a mechanism for retrieve the GUID of an entry from its LN. The Bartender hence uses the Librarian to perform the LN to GUID mapping.

If a GUID exists for the provided LN, then a token corresponding to the command will be put on the place `GUIDexists`; if no GUID exists for LN, then a token will be put on the place `GUIDnotexists`. A central abstraction in our model

is to distinguish between commands which *creates* new entries in the file system (`putFile` and `makeCollection`) and commands which *access* an existing entry (for example `getFile` and `listCollection`). Abstracting from the concrete data stored in the file system implies that commands within each of these two classes can be unified as they have equivalent behaviour from a logical perspective. If the GUID exists, the `CheckAccessCommand` transition checks whether the command provided is an access command and prepares to execute the command by placing a token on place `AccessStart`. Otherwise a failure is indicated to the user by putting a token on the `ToUser` place as determined by the case-expression on the arc from transition `CheckAccessCommand` to place `ToUser`. Similarly, if the GUID does not exist, then the transition `CheckCreateCommand` checks that the provided command is a command which creates an entry; otherwise it issues an error to the user.

The actual execution of access and create commands is modelled by the substitution transitions `AccessEntry` and `CreateEntry`, respectively. Figure 4 depicts the submodule associated with the substitution transition `CreateEntry`. The port place `CreateBegin` (top) is linked to the socket place `CreateStart` in Fig. 3. To create a new entry, its parent collection should exist. This check is modelled by the transition `CheckParentCollection` which also illustrates another important abstraction in our model. The Boolean variable `OK` appearing in the arc expression on the arc from `CheckParentCollection` to place `ParentCollectionExists` (and also in the arc expression on the arc from `CheckParentCollection` to `ToUser`) can be bound to either `true` or `false` when the transition `CheckParentCollection` occurs. This models the two possible outcomes of checking whether the parent collection exists: if `OK=true` then it corresponds to the case where the parent collection exists and a token is put on the place `ParentCollectionExists` according to the then-branch of the if-then-else expression on the arc from `CheckParentCollection` to `ParentCollectionExists`; if `OK=false` an error will be issued to the user by putting a token on the place `ToUser` according to the else-branch of the if-then-else expression in the arc from `CheckParentCollection` to `ToUser`. The non-deterministic choice between the two possible *bindings* of the Boolean variable `OK` hence allows us to model the possible outcomes of checking whether the parent collection exists without a detailed modelling of how this outcome is computed based upon data stored in the system.

If the parent collection exists, the new entry can be created only if the user issuing the command has the right credentials. This check of authorisation is modelled by the transition `CheckAuthorisation` using a similar abstraction as for the check of parent collections described above. If the user is authorised to modify the parent collection, a token will be added to the place `AccessGranted`. Otherwise, a token will be added to the port place `ToUser` to signal an error.

If authorisation is granted, then the actual creation of the entry will be handled by the submodule of the substitution transition `CreateNewEntry`. The submodule of this substitution transition models the creation of a new entry in the database and the addition of this new entry to the parent collection. If this is successful, a token is added to the place `EntryCreatedandLinked`. Otherwise, a

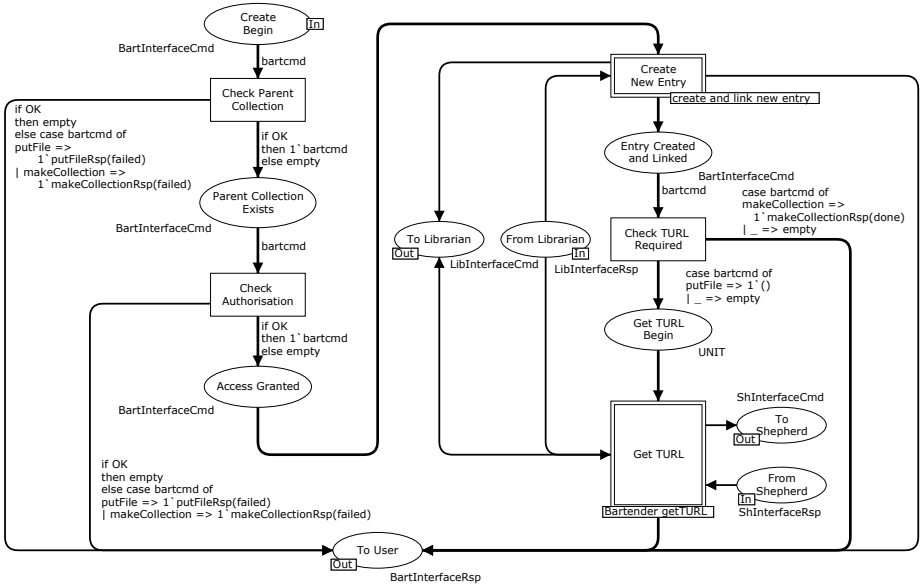


Fig. 4. The CreateEntry submodule

token is added to `ToUser` signalling this error. Once the entry is created and added to the parent collection, we need to determine if a *transfer URL* (TURL) is needed: if the command was `makeCollection`, no file needs to be transferred to the user. If the command was `putFile`, a TURL should be obtained allowing the user to provide the system with the actual file associated to the entry. The transition `CheckTURLRequired` will add a token to `ToUser`, if the command was `makeCollection`. If the command was `putFile`, the transition `CheckTURLRequired` will add a token to the place `get TURL`. The submodule of the substitution transition `GetTURL` models the steps required to retrieve a TURL from the Shepherd. Eventually, this submodule will add a token to the port place `ToUser` to model the sending of the TURL to the user.

We have presented how the `putFile` and `makeCollection` commands have been modelled. The model captures the symmetry between the two commands by using the same transitions and places to model them. The two commands differ only from the transition `CheckTURLRequired`. With the use of a graphical representation, this difference appears explicitly.

## 4 Command Validation

The CPN model of the Chelonia systems was validated by means of simulation and explicit state model checking [3]. The overall purpose of the validation was to establish that the interaction between services in executing a command lead to proper termination of the command. During the construction of the model,

the support for *interactive simulation* in CPN Tools was used to perform detailed checks to ensure that the model behaviour was as desired. An interactive simulation is similar to single-step debugging. During an interactive simulation, the next step is manually selected among the enabled transitions in the current state, and it is possible to observe the effects of the individual transitions directly on the graphical representation of the CPN model. Even though interactive simulations (and simulation in general) do not guarantee correct behaviour, it proved to be very useful in identifying modelling errors.

The final step was to conduct an exhaustive investigation of the model behaviour using *explicit state model checking*. The basic idea of state spaces is to compute all reachable states and state changes of the model and represent these as a directed graph, where nodes represent states (markings) and arcs represent occurring events (transitions). State spaces can be constructed fully automatically by the state space tool of CPN Tools. Validation of the Chelonia system by means of state spaces relied primarily on the use of the *state space report* which can be generated automatically by CPN Tools. The state space report provides information about state space size and standard behavioural properties.

The state space report is divided into several sections, and in the following we explain how the results contained in the report were used to establish the proper termination for the execution of commands. As we have abstracted from the data stored in the file system (as explained in the previous sections) and consider only the control flow of the operations, there is no direct interaction between execution of commands. This means that we can verify each of the commands provided by Chelonia in isolation. This in turn is central in order to alleviate the inherent state explosion problem in explicit state model checking. In the following, we consider the putfile command as a representative example of validation. Validation of the other commands was done in a similar manner.

The state space generated when the model was initialised with the putfile command has 43 states (nodes) and 57 arcs and could be generated in less than a second. To investigate termination properties, we consider the part of the state space report specifying *home and liveness properties*. The liveness properties show that there are two *dead markings* numbered 18 and 43. A dead marking is a state without enabled transitions, and hence represents a state where the execution of the command has terminated. To obtain information about the states 18 and 43, they were transferred into the simulator of CPN Tools and displayed graphically on the CPN model. It was then checked (by inspecting the markings of the places) that state 18 corresponds to a state in which execution of the operation has failed, e.g., due to an authentication error (see Fig. 4). State 43 corresponds to a state in which the putfile command has been successfully executed. Both nodes represent desired possible terminating states for the putfile command execution. Hence, if the putfile command terminates then it terminates in a desired state. Using a state space query it was also shown that the two dead markings constitutes a *home space*. A home space is a set of states with the property that from any reachable state it is always possible to reach at least one state in the set. That markings 18 and 43 constitute a home space means that

we cannot have execution sequences that cannot be extended to reach one of the two markings. Hence, the system design has the property that it is always possible to terminate the putfile command correctly.

The state space report also showed that the strongly connected component graph (SCC-graph) for the state space had 38 nodes and 45 arcs. The fact that there are fewer nodes in the SCC-graph than in the state space implies that there are non-trivial SCCs and hence there are cycles in the state space. Such cycles corresponds to infinite execution sequences. To determine the circumstances under which the execution of the putfile command may not terminate, we consider the fairness properties of transitions which are also provided in the state space report. The fairness properties show that the transition corresponding to servicing a request to the A-Hash is *impartial*. Hence, if the execution does not terminate this is because the systems keeps requesting meta-data from the A-Hash. The request for meta-data is done by the Librarian which may do an arbitrary number of meta-data lookups to resolve the LN. The number of meta-data lookups depends on the length of the LN. Since the LN has a finite length, the number of meta-data lookups is finite. Moreover, impartiality of the transition modelling the servicing of an A-Hash request implies that if the Librarian only does a finite number of request to the A-Hash in the course of command execution, then the execution of the putfile command will terminate. This establishes that with a finite number of lookups in the A-Hash, the execution of the putfile command will eventually terminate (impartiality) in a desired state (acceptable dead markings).

## 5 Conclusion and Future Work

We have presented our CPN-based modelling approach enabling the functional validation of the commands supported by the Chelonia distributed storage system. The main idea in our formal modelling approach was to abstract from the concrete data being stored by the file system, and focus on the control flow in the implementation of the commands provided by the system. This data-independent modelling approach in turn enabled application of finite-state model checking techniques approach for validating the termination properties of the file system commands. The data-independent modelling approach furthermore factored out interdependencies between the commands allowing us to perform the validation of the functional behaviour of each command in isolation. This effectively eliminated the presence of state explosion in the verification of the system. Finally, the CPN modelling language allowed us to structure the model of the Chelonia system which relies on a number of subservices into a hierarchically related set of modules which made both the overall service architecture as well as the control flow of the operations explicit in the structure model.

The main application of the CPN model presented in this paper has been a formal specification of the operations supported by the Chelonia system and its formal validation. One direction of future work is the investigation of how the constructed CPN model can serve as a basis for automatically generate

an implementation of the services of the Chelonia system. The current CPN model is too abstract to be directly used, and hence needs to be detailed in a series of formal refinement steps. These refinement steps need to include a precise description of data structures used in Chelonia while the model still needs to remain abstracted from the actual data in order to facilitate verification. Such a refinement of the model will also allow the verification of properties related to the access right management depending on meta-data entries in the system.

*Acknowledgement.* This work has been funded by the Norwegian Research Council (NFR) project 194521 (FORMGRID).

## References

1. Andreozzi, S., Sgaravatto, M., Vistoli, M.C.: Sharing a Conceptual Model of Grid Resources and Services. CoRR, cs.DC/0306111 (2003)
2. Bratosin, C., van der Aalst, W.M.P., Sidorova, N.: Modeling Grid Workflows with Colored Petri Nets. In: Proc. Workshop on the Practical Use of Coloured Petri Nets and CPN Tools, pp. 67–86. Aarhus University - DAIMI-PB 584 (2007)
3. Clarke, E., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge (1999)
4. CPN Tools Homepage, <http://www.cpnertools.org>
5. Dalheimer, M., Pfreundt, F., Merz, P.: Formal Verification of a Grid Resource Allocation Protocol. In: Proc. of CCGRID 2008, pp. 332–339. IEEE, Los Alamitos (2008)
6. Domenici, A., Donno, F.: Static and Dynamic Data Models for the Storage Resource Manager v2.2. Journal of Grid Computing 7(1), 115–133 (2009)
7. Li, B., et al.: A Formal Model for the Grid Security Infrastructure. In: Zhou, X., Su, S., Papazoglou, M.P., Orlowska, M.E., Jeffery, K. (eds.) WISE 2004. LNCS, vol. 3306, pp. 706–717. Springer, Heidelberg (2004)
8. Badino, P., et al.: The Storage Resource Manager Interface Specification v2.2. Technical report, Lawrence Berkeley National Laboratory (2009)
9. Andreozzi, S., et al.: GLUE Schema Specification version 1.3. Technical report, INFN - Istituto Nazionale di Fisica Nucleare, Italy (January 2007)
10. Industrial Use of CPNs, <http://www.cs.au.dk/CPnets/intro/example/indu.html>
11. Hlaoui, Y., Benayed, L.: Symbolic Model Checking Supporting Formal Verification of Grid Service Workflow Models Specified by UML Activity Diagrams. In: Proc. of NOTERE 2010, pp. 255–260. IEEE Computer Society, Los Alamitos (2010)
12. Jensen, K., Kristensen, L.M.: Coloured Petri Nets – Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009)
13. Lai, H.F.: Modeling Grid Workflow by Coloured Grid Service Net. In: Bellavista, P., Chang, R.-S., Chao, H.-C., Lin, S.-F., Sloot, P.M.A. (eds.) GPC 2010. LNCS, vol. 6104, pp. 204–213. Springer, Heidelberg (2010)
14. Manset, D., Verjus, H., McClatchey, R., Oquendo, F.: A Formal Architecture-Centric Model-Driven Approach for the Automatic Generation of Grid Applications. CoRR, abs/cs/0601118 (2006)
15. Mascheroni, M., Farina, F.: Nets Within Nets Paradigm and Grid Computing. In: Proc. of PNSE 2010, pp. 23–38. Universitat Hamburg - FBI-HH-B-294/10 (2010)

16. Chelonia CPN Model, <http://www.hib.no/ansatte/lmkr/cpnchelonia.xml>
17. Nagy, Z., Nilsen, J., Toor, S.Z.: Chelonia - Self-healing Distributed Storage System – NorduGrid Technical Report 17 (2010), <http://www.knowarc.eu/chelonia/>
18. Nagy, Z., Nilsen, J., Toor, S.Z.: Chelonia - Self-healing Distributed Storage System - Documentation of the ARC Storage System. Technical report, NORDUGRID (February 2010)
19. Reisig, W.: Petri Nets. EATCS Monographs on Theoretical Computer Science, vol. 4. Springer, Heidelberg (1985)
20. Ullman, J.D.: Elements of ML Programming. Prentice-Hall, Englewood Cliffs (1998)

# An Integrated Network Scanning Tool for Attack Graph Construction

Feng Cheng, Sebastian Roschke, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam, 14482, Potsdam, Germany  
{feng.cheng,sebastian.roschke,christoph.meinel}@hpi.uni-potsdam.de

**Abstract.** Scanning is essential for gathering information about the actual state of computer systems or networks. Therefore, it is always taken as the first step of potential attacks against targets. In certain cases, scanning itself is categorized as an attack. Scanning can on the other hand be used for the right purposes, for example, checking the system configurations, verifying firewall rules, proofing security polices, as well as monitoring the large scale network environment. From this point of view, scanning is an effective method for system or network management, security measurement and auditing. To visualize, analyze, and finally evaluate the data gathered by scanners, Attack Graph plays an important role. High quality information about the target system or network is the prerequisite for constructing the attack graph. However, different implementations of scanners have different capabilities and always result in different kinds of outputs. These outputs are usually heterogeneous and not machine-readable, which makes the further analysis a challenging task. In this paper, we examine common types of scanners and demonstrate how to combine multiple types of scanners. The results of all the involved scanners are integrated into a well-designed and consistent data structure, which can not only be well interpreted by human security specialists but also be directly fed into an attack graph construction tool.

## 1 Introduction

Scanning is always the first step towards successful exploits of the target system [1], [2]. Normally, hackers are not able to know anything about the targets without scanning. Gathering information about the adversary is the unique start point. Most complex attacks are practically impossible without continuous scanning on the final target. On the other hand, the scanning techniques have also been adopted as an important approach for security experts or network administrators to check the weaknesses of target systems or networks [3], [4]. Modern IT infrastructures become more and more sophisticated. Failures due to both technical and human factors, might take place in the processes of design, implementation, deployment, configuration, and execution. Carrying out black-box mode tests is always required for detecting such failures, especially the security vulnerabilities. Scanning is, in most cases, an effective tool for such tests [5]. Along with the increased requirements from both attackers and defenders, a large number of scanning techniques have emerged in recent years, e.g.,



*nmap*<sup>[1]</sup>, *amap*<sup>[2]</sup>, *netcat*<sup>[3]</sup>, *P0f*<sup>[4]</sup>, *XProbe*<sup>[5]</sup>, *X-Scan*<sup>[6]</sup>, etc. Different types of scanning techniques have different competencies and therefore result in different kinds of implementations. The outputs of those scanners are usually heterogeneous and not machine-readable. Combining multiple types of scanning techniques to get comprehensive but easy to interpret scanning results is expected. Attack Graph offers a formal method to represent the ways in which an attacker can exploit vulnerabilities and break into a system [6], [7]. To construct an attack graph, the accurate information about the target system or network is required. Scanning techniques play important roles in this context. How to integrate scanning tools into the automatic attack graph construction platform is a challenging task for creating high quality attack graphs.

In this paper, we propose an integrated network scanning tool by combining multiple types of scanners. The results of all the involved scanners are integrated into a well-designed and consistent data model, which can not only be well interpreted by human security specialists but also be directly fed and matched with the vulnerability representation within the attack graph tools [8].

This paper is organized as follows. Section 2 introduces some related works, i.e., the workflow of the Attack Graph construction and some popular scanning techniques. Section 3 proposes our integrated scanner. A practical scanning example is shown as proof-of-concept in Section 4. We conclude the paper in Section 5.

## 2 Related Work

### 2.1 Attack Graphs

Attack Graphs have been proposed for years as a formal way to simplify the modeling of complex attacking scenarios [9]. An attack graph describes not only one possible attack, but also many potential ways for an attacker to reach a goal. The workflow of an attack graph construction tool consists of three independent phases: Information Gathering, Attack Graph Construction, as well as Visualization and Analysis. In the information gathering phase, all necessary information to construct attack graphs is collected and unified, such as information on network structure, connected hosts, and running services. In the attack graph construction phase, a graph is computed based on the gathered system information and existing vulnerability descriptions. Finally, the attack graph is processed in the visualization and analysis phase. Attack graphs always require a certain set of input information. For one, a database of existing vulnerabilities has to be available, as without it, it would not be possible to identify or evaluate the effects of host-specific weaknesses. Also, the network structure must be

---

<sup>1</sup> <http://nmap.org/>

<sup>2</sup> <http://freeworld.thc.org/thc-amap/>

<sup>3</sup> <http://netcat.sourceforge.net/>

<sup>4</sup> <http://lcamtuf.coredump.cx/p0f.shtml>

<sup>5</sup> <http://sourceforge.net/projects/xprobe/>

<sup>6</sup> <http://xfocus.net/>

known beforehand. It is necessary to identify which hosts can be reached by the attacker. Often, a network scanning and host-based vulnerability analysis are performed before the attack graph is constructed [8].

## 2.2 Review of Scanning Techniques

We can classify the scanning techniques into the following categories:

**Passive.** No packages sent at all.

**Discovery.** Packages are only sent to find hosts and detect the connectivity.

**Port Scan.** Systematic testing of individual ports or ranges. Response packages might be examined for software banners.

**Probing.** Discovered services are checked for versions and vulnerabilities.

**Exploiting.** Vulnerabilities are actively used or verified.

**Other.** It is active but cannot be classified as any value above.

Most of emerged scanning techniques are implemented based on this classification. In practice, it includes:

**Topology Scanner** is used to investigate the structure of networks, which encompasses the detection of hosts residing in a network, the network mask, and the gateways. For advanced topology scanners, multiple networks are scanned and then the results can be put together into a complete network diagram for evaluating their interconnections. The participants in a local network can be obtained by broadcasting *ping* or *ARP*-request packets. For remote hosts, ordinary *ping* over a large address range can be performed. Popular implementations include *traceroute*<sup>7</sup>, *Nessus*<sup>8</sup>, etc.

**Port Scanner** aims at finding open and filtered ports on a specified target host. There have been implemented a variety of such scan methods, e.g., *Nmap*, *Nessus*, *netcat*, etc. Most of them are targeted for TCP ports. The easiest and also the most obvious is a *connect* scan. Stealthier methods are *syn* or *ack* scans. UDP scans are used for UDP-based ports.

**Fingerprint Scanner** is to find the name and version of the software or operating system running on the target. The primary method to obtain product information from a service is via banner grabbing. Advanced tools examine the specifics of sent packets and the behavior the network stack. Known examples are *amap*, *P0f*, *XProbe*, etc.

**Vulnerability Scanner** targets at detecting vulnerabilities of services or the operating systems running on a host. Vulnerabilities are usually found by actually trying to exploit a set of possible vulnerabilities. Less aggressive tools infer possible vulnerabilities from the products determined by fingerprinting. Existing vulnerability scanners include *Nessus*, *X-Scan*, *Nmap*, etc.

**Incremental Scanner** usually scans the surrounding network topology by first performing a scan from any of the previously mentioned categories and gain

<sup>7</sup> <http://www.traceroute.org/>

<sup>8</sup> <http://www.nessus.org/>

access to vulnerable hosts with exploits. Having access to further hosts, already found networks can be scanned from another viewpoint or completely new networks may become accessible. Doing this incrementally, it sometimes enables the mapping of DMZs (demilitarized zones) between a public and private network or even internal private network. Typical methods to accomplish incremental scanning is to exploit a vulnerability on a host and inject scanning code into the vulnerable service. Some common approaches are *syscall proxying* [1] and *SNAPP* [10], *Core Impact* [1], etc.

### 3 An Integrated Network Scanning Platform

Each scanner delivers a dedicated set of information about the target host or network. To obtain a rather complete picture of a target for more accurate post processing, e.g., constructing attack graphs, it requires the combination of results from multiple scanners. However, the handling of multiple scanners is not easy due to the following two reasons:

**Configuration.** Along with the advanced usability of the modern scanner, the required configuration effort becomes more and more complex. For example, both *Nmap* and *Nessus* provide an overwhelming number of configuration options. Consequently, it takes users much more effort to learn how to implement a proper and efficient scanning operation.

**Output Interpretation.** Each scanner outputs its results in a certain format. Unfortunately, this format is not unified. A user needs to familiarize with each format individually. Furthermore, some detailed scan results are only written in a report file which might be even harder to read.

To achieve an abstraction from all the sub-scanners, following challenges need to be considered for the implementation of the integrated scanner platform.

1. The integrated sub-scanners should be transparent to the user, which means that the user does not need to know how to use the sub-scanners and even does not actually recognize their involvements.
2. The platform must be able to control the actions of all sub-scanners. As different scanners significantly differ in the functionalities, the configuration can be quite different as well. The challenge is to find a common set of configuration options.
3. The scan results of network scanners are not unified in any form. They might be available as XML files, graphically on a GUI or text on the command line output. In order to provide a consistent scan result, output of different forms must be converted into one common result format.
4. As many sub-scanners are used within the integrated platform, it is highly possible that a certain type of scan result is already available from another scanner. In this case, the duplicate information might exist. We need indicate the same result or, in the worst case, contradicting information.

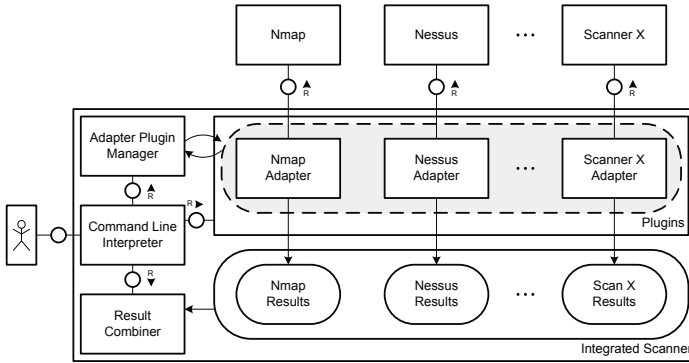


Fig. 1. Architecture of the *Integrated Network Scanning Platform*

### 3.1 Basic Architecture

Figure 1 shows a rough overview of the architecture used for our proposed *Integrated Network Scanning Platform*. Although the attached scanners do not belong to the platform, they constitute the foundation for all the supported scanning functionalities. Currently, we have focused on online scanning with the popular scanners *Nmap* and *Nessus*. The design of the platform is highly flexible and extensible. It consists of several components. The upper part encompasses multiple plug-ins communicating with the integrated scanners. They basically act as adapters between one scanner and the other components of the platform. These adapters are designed as plug-ins, so that other scanners can be added easily. A plug-in has to accomplish two major tasks to enable seamless integration of scanners into the platform:

1. Bind to an interface of the scanning tool. After binding, the plug-in provides an abstract control interface to the scanner usable by the platform. One part of this interface provision is the translation of the platform's abstract scan options into specific scan options interpretable by the scanner. After an initiated scan has been successfully finished, the plug-in has the task to obtain the scan result from the scanner.
2. Convert the specific scan result obtained from the scanner into a scan result with a unified format, so that the platform only needs to handle one abstract scan result format.

When the scan results have been provided by the scanner adapters, the *result combiner* can start its work by combining all single results into one consistent overall result. The management of the adapter plug-ins is performed by a *plug-in manager* component. It allows to execute plug-ins concurrently as well as pass abstract scan parameters to all adapters. The bridge between the previously mentioned components and the user of the platform is provided by the *command-line interpreter*. It is currently deployed on the host of the scanning user.

### 3.2 Finding a Common Data Model

As previously mentioned, a major challenge for creating such an integrated scanner is to find a common data model for containing the results from different scanners. Such a model should be easy to understand as well as easy to interpret by computers. For being fed to the later attack graph construction tools, the data model should fit to the formal representation of the vulnerability information [8]. *Nmap* offers such a data model for containing its own scanning result. However, it is strictly tailored to the capabilities of *Nmap*'s port scanner and is hard to comprise other types of scanning results. The vulnerability scanner, *Nessus* also provides a data format for representing the revealed information [11], but it focuses more on the analysis and predictions of the vulnerabilities. Here we propose a custom XML file format as shown in Figure 2.

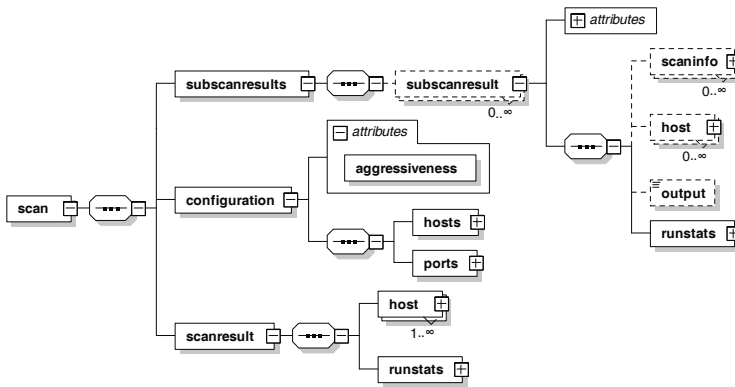


Fig. 2. Top Level Structure of the Data Model

This data model has been divided into three top level branches which cover the different aspects of an integrated scan. One branch contains the combined scan result from all scanners. Another branch contains an abstract configuration used for the integrated scan. In order to determine the origin of a certain piece of information in the combined scan result, the last branch keeps a copy of all individual scan results produced by the scanners. These scan results will be called *sub-scans* in the following. Both the combined result and sub-scan results have a similar format. This format only focuses on one scan result and is very close to *Nmap*'s single result format. A host can be regarded as central element in this format. Each host then has a set of open ports and running services. Beyond that, meta information about the scan can be stored. The sub-scan branch additionally contains detailed information about the scan tool having produced the sub-scan result. This encompasses, for example, tool-specific parameters used as configuration and the original output of the tool. The original output can later be used to obtain information being lost during the conversion to the common data model. The configuration branch stores an abstract view on the parameters being passed to the scanners.

### 3.3 Parsing the Scanner Outputs

The adapter is the key component to convert the specific output from the scanners to sub-scans of our common data model. For *Nmap*, this task is rather straightforward as our sub-scan branch is very similar to *Nmap*'s data model. Only some smaller changes needed to be implemented. One example is the standardization and unification of information specified in *Nmap*'s output. Therefore, time and date information needed to be converted to one common date format and product names were unified by means of the *Common Platform Enumeration* (CPE) [12]. Due to generalization of *Nmap*'s format, we had to omit some information or had to move some specific information into a more general context. As an example for this generalization, the device type for a host had to be inferred from the device types previously specified in the services of hosts. In the case of *Nessus*, the conversion turned out to be more complex. The problem with its XML-based *.nessus.v1* format is plain text which is hard to interpret by a program. In fact, each security issue reported by *Nessus* is inserted into just one XML **data** tag. Thus, we need to make the output interpretable by an extensive use of regular expressions. A new output format called *.nessus.v2* which addresses this issue of only providing textual data has been introduced by *Tenable* in late 2009. Another issue which makes it more complicated to interpret the *Nessus*' is the organization of *Nessus*' own plug-ins. Each aspect of a scan is handled by a certain plug-in. For example, there is a plug-in to list the open ports, another plug-in provides the host name of the target host, and yet another plug-in checks if a specific vulnerability is available on the target. As a consequence, the information has to be collected from multiple plug-in outputs which do not even have a standardized format. Therefore, it is nearly impossible to include all the results from *Nessus* into our data model. Even if it is possible to evaluate the output of all current plug-ins, new plug-ins would result in an incomplete processing sooner or later.

### 3.4 Merging Redundant Information

Although the sub-scan provides information about each single scan, a combined scan result is desirable for clarity. Redundant information being produced multiple times must be merged. Such information can be of two types, i.e., matching or contradicting. Matching information can be easily merged by keeping only one instance. We decided to handle contradicting information by plausibility checks and a combination of voting and scoring mechanisms. The different approaches are listed as follows.

**Voting.** A piece of information is considered as right only if it has been reported by the majority of scanners. This mechanism only makes sense if there are more than two integrated scanners.

**Scoring.** A scanner will be assigned scores corresponding to the reliability of the information it produces.

**Plausibility.** For some types of information, it can be inferred logically which information is more likely the right one.

As shown in Figure 3, the information about the detected operating systems is an example which might be reported differently among scanners. In many cases, even one scanner alone delivers multiple possible operating systems. Fortunately, these scanners also provide a scoring for the most probable entries. Among scanners, we first use a voting and then perform a scoring on ambiguous results. Here, Nessus' results are weighted higher than Nmap's results, as for fingerprint scanning, Nessus is more reliable. Additionally, in the merged result, the accuracy values have a sum of 1, in contrast to the relative accuracy in the individual results.

The example in 4 shows how to remove redundant information about port states using plausibility checks. If multiple states are reported, we choose the entry which first appears in the order *open*, *closed*, and *filtered*. A port with state *open* or *closed* sends packets to the scanner whereas filtered ports do not send any packet. So a filtered port might be the result of a lost packet.

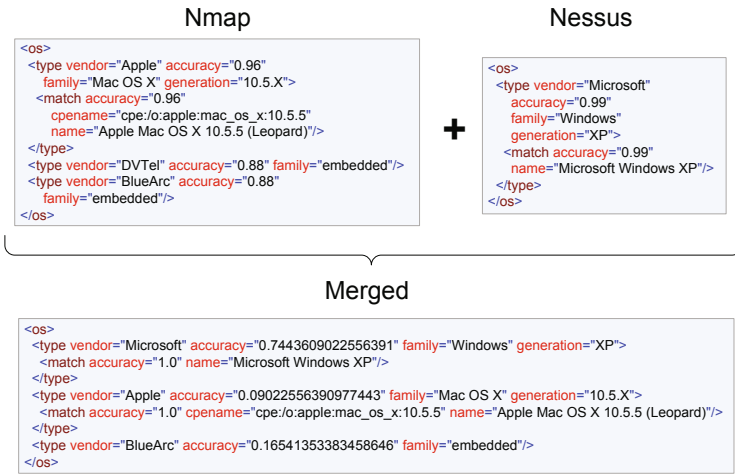


Fig. 3. Merging operating system information from Nessus and Nmap into one consistent overview of the target's operating system

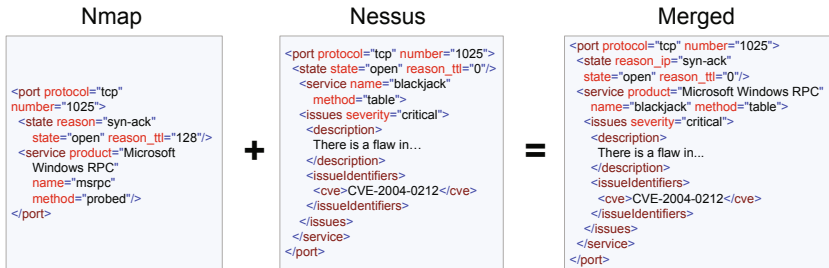


Fig. 4. Merging port information from Nessus and Nmap into one port information

## 4 Scanning in the Practice

To show how this integrated scanner works in the practice, we set up a highly vulnerable virtual machine and perform an integrated scanning against it. We have selected a very outdated Ubuntu distribution, namely *Ubuntu 6.06 Dapper Drake*. It has a 2.6.15 kernel, which bears a high security risk. Additionally, we have installed a multitude of seriously vulnerable network services on the virtual machine, including *ProFTPD 1.2.5*, *Apache HTTP Server 2.0.55*, *Bind 9.3.2*, *Samba3.0.22*, *Courier IMAP (imapd) 3.0.8*, and *MySQL Server 5.0.22*. It is obvious that our integrated scanner can find many open ports and even more vulnerabilities of the installed products. For the practical scan, we have tested two different configurations, i.e., *probing* and *exploiting*. The *probing* mode is used to create static attack graph while *exploiting* mode can gather information required by dynamic attack graph construction.

With the *probing* configuration, we could find all open ports affiliated with most of the installed products, see Figure 5(a). The operating system could be determined as well. However, the scan only reported a *Samba* vulnerability. The reason for this sparse result is the fact that with probing, vulnerabilities are only inferred from examined product versions.

We could gain a better result with the *exploiting* configuration as shown in the Figure 5(b). It delivered about 30 vulnerabilities of which the most were also reported in vulnerability databases. A significant drawback of the *exploiting* configuration is its long runtime which is specifically caused by the *Nessus* sub-scan. It took nearly half an hour to finish.

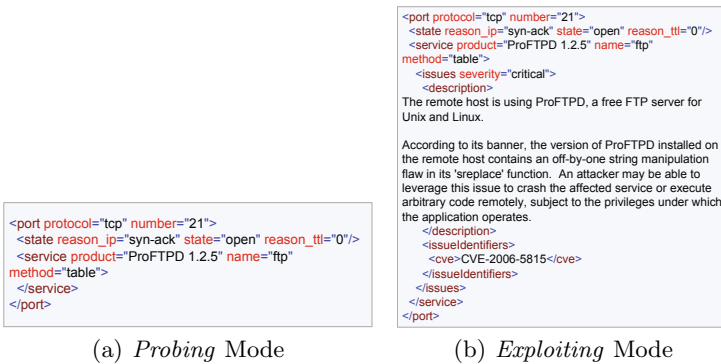


Fig. 5. Information about the exposed ProFTPD service in the scanning results

## 5 Conclusion

Scanning focuses on more or less serious information disclosure and vulnerabilities about the target. It can be accomplished with a variety of scanning tools today. All of them require an individual way of interaction and provide scanning



results in different output formats. As a consequence, it is hard for a user to get along with the multiple scanner tools. We have designed and implemented a prototype of the integrated scanning tool allowing to access scan results from multiple scanner tools with just one user interface in a unified output format. As the integrated scanner is configurable for different levels of scan aggressiveness, it is suitable for different types of attack graph construction tools.

## Acknowledgement

The authors would like to thank David Jäger and Arvid Heise for their support on implementation of the proposed tool.

## References

1. Caceres, M.: Syscall Proxying - Simulating Remote Execution. Technical report, Core Security Technologies (2002)
2. Moore, H., Val-smith: Tactical Exploitation, Version 1.0.0. Technical report, Metasploit LLC (2007)
3. Bhatia, A., Lam, B., et al.: Automated Network Security Audit Tool (ANSAT). Technical report, University of Colorado at Boulder (2006)
4. Bishop, M.: About Penetration Testing. *IEEE Security & Privacy* 5, 84–87 (2007)
5. Siamwalla, R., Sharma, R., Keshav, S.: Discovering Internet Topology. Technical report, Cornell University (1998)
6. Dawkins, J., Clark, K., Manes, G., Papa, M.: A Framework for Unified Network Security Management: Identifying and Tracking Security Threats on Converged Networks. *Journal of Network and Systems Management* 13(3), 253–267 (2005)
7. Schneier, B.: Attack Trees - Modeling Security Threats. *Dr. Dobbs's Journal* 21, 21–29 (1999)
8. Cheng, F., Roschke, S., Schuppenies, R., Meinel, C.: Remodeling Vulnerability Information. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) *Inscrypt 2009*. LNCS, vol. 6151, pp. 324–336. Springer, Heidelberg (2010)
9. Jajodia, S., Noel, S., O'Berry, B.: Topological Analysis of Network Attack Vulnerability. In: Vipin Kumar, J.S., Lazarevic, A. (eds.) *Managing Cyber Threats: Issues, Approaches, and Challenges*. *Massive Computing*, vol. 5, pp. 247–266. Springer, Heidelberg (2005)
10. Cheng, F., Wolter, C., Meinel, C.: A Simple, Smart and Extensible Framework for Network Security Measurement. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) *Inscrypt 2007*. LNCS, vol. 4990, pp. 517–531. Springer, Heidelberg (2008)
11. Arboi, M.: *The NASL2 Reference Manual*. Tenable Network Security (2005)
12. Cheikes, B.A., Waltermire, D.: *Common Platform Enumeration: Naming Specification Version 2.3 (DRAFT)*. Technical Report The MITRE Corporation, National Institute of Standards and Technology, NIST (2010)

# Context-Awareness Micro-architecture for Smart Spaces

Susanna Pantsar-Syvaniemi, Jarkko Kuusijärvi, and Eila Ovaska

VTT Technical Centre of Finland, P.O. Box 1100,  
90571 Oulu, Finland  
{Susanna.Pantsar-Syvaniemi, Jarkko.Kuusijarvi,  
Eila.Ovaska}@vtt.fi

**Abstract.** Reaching context-awareness is still an open issue in pervasive systems like smart spaces (SS). With the help of context a smart space application can react to the current situation or be proactive and take into account coming circumstances. This paper describes the ongoing research on how context-awareness - an ability to identify and react - is enabled for the heterogeneous smart space applications. This work presents innovative context-awareness micro-architecture to design software in a way that it is able to react to changes based on information gathered via a context monitoring agent and inferred by a context reasoning agent. The usage of the context-awareness micro-architecture is validated with two scenarios which are instantiated to a personal smart space and a smart home.

## 1 Introduction

A smart space can be any of our everyday environments that are enriched with smart functionality. The smart space can be, e.g., a personal space built around a user and her mobile device, a smart car, a smart home shared with her family, friends and other persons like a window cleaner; a smart office; and a smart city. Because the smart spaces are heterogeneous they vary a lot, e.g., in the viewpoint of 1) an amount of information produced and consumed in it; 2) a privacy of the smart space, and 3) a security and trustworthiness of the information. For the smart space applications it is challenging to model the information and to build up the needed functionalities for monitoring, reasoning and performing adaptation based on the deduction made. That is due to continuously evolving smart space applications where the information can not be predefined and the information is really heterogeneous because of the numerous users and different kinds of devices interacting in the smart space. The devices can be following different standards, fabricated by various manufactures and they are also having different life cycles.

Hong et al. [1] represent (based on their literature review) that context-aware systems 1) are still developing in order to improve, and 2) are not fully implemented in real life. They also highlight that research on design patterns of context-aware systems should be conducted because they will provide an effective way for sharing solutions to design problems and reuse prior design knowledge. Bettini et al. [2] highlight that due to the inherent complexity of context-aware applications, the development should be supported by adequate context information modeling and reasoning

techniques. Truong and Dustdar [3] explain that distributed context management, context-aware service modeling and engineering, context reasoning and quality of context, security and privacy, have not been well addressed in the Context-aware Web Service Systems. Indulska and Nicklas [4] point out that development of context-aware applications is complex as there are many software engineering challenges stemming from heterogeneity of context information sources, imperfection of context information, and the necessity for reasoning on contextual situations that require application adaptations.

In this work we present an innovative and novel context-awareness micro-architecture that answers to the need of applying design patterns for context-aware systems. In addition, the context-awareness micro-architecture aims to tackle the challenges relating to 1) the heterogeneity of context information handling, 2) reasoning on real world situations, and 3) doing adaptations based on the current and historical context information. According to Guéhéneuc [5] micro-architecture is concrete manifestation (representation) to be followed in the implementation of a system, describing the design motives by the means of composed of classes, methods, fields, and relationships having structure and organization similar to one or more motives. We utilize the context-awareness micro-architecture for describing a reusable structure that follows the well-known phases of adaptive control systems - monitoring, reasoning and adaptation. This means self-adapting systems, where software evaluates its own behavior and changes it when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible [6].

The structure of the paper is as follows. Section 2 presents the background. Section 3 introduces our context-awareness micro-architecture. Thereafter, Section 4 goes through the validation where the context-awareness micro-architecture is applied for the development of the cross-domain scenario between the personal space and the smart home. Conclusion and future work close the paper.

## 2 Background

To catch the smartness for a space around person(s), the information needs to be modeled, i.e., the context of the devices and the person(s). The context can be used, e.g., 1) for adapting behavior of an application according to resources in the execution environment or user needs and preferences; and 2) for managing quality at run-time. The highly used definition for context is: ‘Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and application, including the user and applications themselves’ [7].

Eugster et al. in [8] present the middleware classification they performed for 22 middleware platforms from the viewpoint of a developer of context-aware applications. Only few middleware platforms (3 platforms out of 22) were interesting for us because they provide both the high-level programming support and comply with the three given architectural dimensions. The high-level programming support means that the middleware platform adds a context storage and management layer to the previous layers. The three architectural dimensions are: (1) decentralization, (2) portability and

(3) interoperability. Decentralization measures a platform's dependence on specific components. Portability classifies platforms in two groups: portable platforms can run on many different operating systems, and operating system dependant platforms, which can only run on few operating systems (usually one). Interoperability then measures the ease with which a platform can communicate with heterogeneous software components. Ideal interoperable platforms can communicate with many different applications, regardless of the operating system they are built on or of the programming language they are written in. Those three middleware platforms were CARISMA, Hydrogen, and CoWSAMI. Mostly, they do not have ontology support to provide an information-based interoperability as is available in an interoperability platform (IOP) developed in the SOFIA-project [9].

In this work, the smart space applications are designed by using the IOP. For the implementation we utilized the Smart-M3 platform [10] that is one realization of the IOP. Therefore, we use the related terminology. The purpose of the IOP is to make the information available between heterogeneous smart objects (devices) and add semantics for this information. In the IOP, a SIB (Semantic Information Broker) creates a backbone to the smart space. In the SIB all the information is based upon the idea of making statements in the form of subject-predicate-object expressions. These expressions are known as triples in the Resource Description Framework (RDF). A KP (Knowledge Processor) works as an independent agent producing and/or consuming information from the SIB. Thus, all the information is transmitted via the SIB, i.e., KPs do not communicate directly with each other. The information transmission is performed by a SSAP (Smart Space Application Protocol). The SSAP can be utilized with different communication protocols, e.g., TCP/IP and Bluetooth.

In our earlier work, as introduced in [11], we propose that "A context defines the limit of information usage of a smart space application". That is based on the assumption that any piece of data, at a given time, can be context for a given smart space application. The work presented here is using a context-awareness concept besides the IOP. The context-awareness concept is our previous work and was introduced in [12]. That concept consists of novel context ontology and software agents for context monitoring, reasoning and context-based adaptation. Software agents are called KPs in the IOP. The context-awareness concept is based on the semantic context information triangle presented by Bettini et al. [2]. In the context-awareness concept the conceptual context ontology is defined with three contextual levels: a physical context of environment, a digital context of environment and a situation context. Fig. 1 presents the context-aware agents on a contextual level to which they mostly belong to. Thus, context monitoring agents are on the lowest level in the contextual hierarchy. Context reasoning agents use the context info provided by the lowest level and produce new, inferred context info to be utilized by context-based adaptation agents. Our context ontology is gathered based on the context dimensions for the physical context of the environment, the digital context of the environment and the human context. We selected the ontology-based model and OWL (Web Ontology Language) for describing the context ontology. The reason for our selection was due to OWL support for interoperability and heterogeneity that are needed for ontology to be able to evolve in the future. Our novel context ontology exploits some parts from SOUPA [13] and the upper context conceptualization presented in [14]. Thus, the context-awareness concept is utilized as a basis for the context-awareness micro-architecture presented in

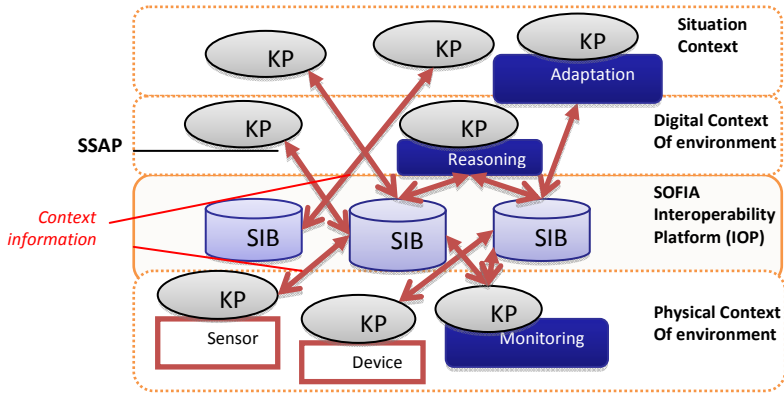


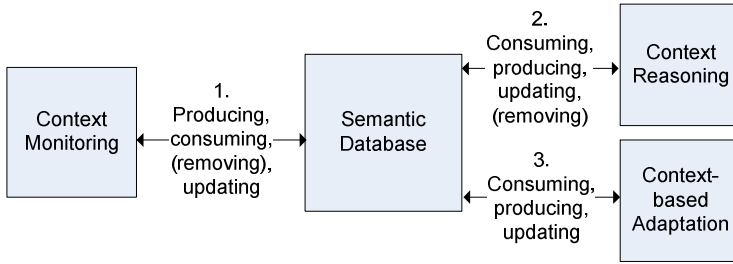
Fig. 1. Context-awareness concept

this paper. We follow also a separation of concerns principle to keep application logic free of context. Therefore, the context-aware agents are illustrated in Fig. 1 as own entities separated from the SIB and applications.

The context monitoring agent has been defined to be used for the security adaptation in the smart space application as is reported in our previous work, in [15]. That work presents an adaptation approach that offers required flexibility and autonomy especially for smart spaces – by utilizing context monitoring and security measuring. A context change is a trigger for the security adaptation in the smart space application. The context change means either a change in the physical or digital environment of an application or a change in the usage of the application. Based on these changes the application makes a decision on its required security level. After that, the application (or supporting services) measures whether the requirement is met or not. Finally, the application adapts the used security mechanisms and parameters if the desired security level cannot be met with the currently used mechanisms.

### 3 Context-Awareness Micro-architecture

In this section we introduce the context-awareness micro-architecture which can be used as a reusable solution with agents that can be updated dynamically. The context-awareness micro-architecture consists of three types of agents: the context monitoring, the context reasoning and the context-based adaptation agents. The emphasis in this work is on the reusable solution with agents and because of that the context ontology is kept in the minor role. The context ontology is presented in our previous work [12]. Fig. 2 illustrates the structural viewpoint of the logical context-awareness micro-architecture. The agents do not communicate directly between each other; instead they have an interface with the semantic database. (In the IOP and in the Smart-M3 Platform the semantic database is known as the SIB.) That interface is mainly used for producing and consuming the context information in the pervasive systems like smart spaces. Of course, the interface can be used for removing and updating the produced information. Hence, context-awareness agents are tied together via the semantic database. The execution order of the agents is shown in Fig. 2 with numbered



**Fig. 2.** The logical structure of the context-awareness micro-architecture

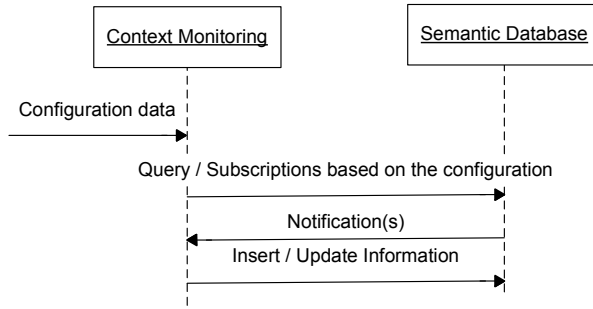
connections. Firstly, the context monitoring agent provides information to the SIB to be used by the context reasoning agent. Lastly, the context-based adaptation is notified by the information updated by the context reasoning agent.

The ultimate idea is that all the agents from the context-awareness micro-architecture are realized for the smart space, although that is not mandatory. The realization or localization is decided by the smart space creator and administrator (might be an owner as well). The localizations vary at least based on the type of the smart space, e.g., is it a personal, a home, an office, or a city space. Hence, in some localization the adaptation might be included to the context reasoning agent. That is possible because both the reasoning agent and the context-based adaptation agent are guided via rules. The rules can be set at design-time or they can be given by the user or application at run-time or when the context-awareness agent is activated. The rules will be saved to the SIB as text strings to enable their updating at run-time. The context-awareness agents will need a parser to interpret the rule strings. We will structure the agents from the context-awareness micro-architecture so that each agent will be a reusable software component. These agents can be deployed separately to the different computing environments because they communicate only on the information level via the SIB. Next we will introduce the agents, i.e., the software components.

### 3.1 Context Monitoring

The context monitoring agent will be configurable and reusable. It can be thought as a design pattern that can be used to build monitoring functionalities to the smart space application. The context monitoring agent can be configured by the rules on the design-time, on the starting phase of the smart space application or at run-time. On the design-time the relevant rules are created by the designer. On the starting phase the configuration (rules) will be given with, e.g., a configuration file when the context monitoring agent is activated. At run-time the rules will be updated via the SIB. Based on the needs many instances or localizations from context monitoring agents can be created to the one smart space application.

The context monitoring agent will be configured to collect or monitor certain information that is relevant context, e.g., for security monitoring or energy consumption. Based on the configuration the context monitoring agent makes subscriptions to the information in the SIB. It can also query the relevant information from the SIB. The context monitoring agent can be utilized, e.g., in the user interface to collect



**Fig. 3.** Example sequence diagram of the context monitoring agent

preferences from the home owner or from the owner of the smart space. In Fig. 3 we introduce more specialized context monitoring agent which is able to filter the certain situation from the monitored information. It will insert or update information to the SIB when the situation has occurred. The context monitoring agent has already been utilized as a building block for the security adaptation as above mentioned. The usage as a building block exemplifies the easy adoption of the context monitoring agent.

### 3.2 Context Reasoning

The context reasoning agent will be utilized for reasoning based on the requirements set in the form of rules. The rules can be set at design-time or they can be given by the user or application at run-time or when the context reasoning agent is activated; see Fig. 4. Thus, the context reasoning agent is configurable. In the smart space we can have many variations of the context reasoning agent, e.g., one for reasoning energy consumption at home and other one for brewing coffee at right time on the morning, and third one for taking care of the smart lighting at the home, in the garden, and in the garage. The context reasoning agents may use both historical and additional context information. The context reasoning agent can be divided into analyzing and reasoning parts when reasoning is complex.

### 3.3 Context-Based Adaptation

The context-based adaptation agent is used for adapting the smart space applications based on 1) the current context or 2) both the current and the historical context. The current context means the application's situation at the present moment. The historical context is saved to the SIB to be utilized later. The current context will be limited and inferred by the monitoring and reasoning agents before it is available for the context-based adaptation agent. The context-based adaptation agent is acting on the uppermost context level as shown in Fig. 1. The context-based adaptation triggers the needed actions to finalize the situation in question. With the situation we mean "the certain state in the smart space" that is quite similar than what is meant with the situation, "a situation is a temporal state within context", in [2]. In the simple situation the adaptation can be something really straightforward; to switch on, e.g., the lights in an entrance hall when the front door is unlocked. The adaptation means that the wanted

action or functionality is activated based on the inserted or updated triple in the SIB. The “functionality triple” is introduced to the smart space according to the used ontology. Depending from the rules set by the smart space owner (or definer) the context-based adaptation can include a learning part, too. The rules can be brought to this agent in the same way as for the context reasoning agents. The behavior of the context-based adaptation agent is shown in Fig. 5.

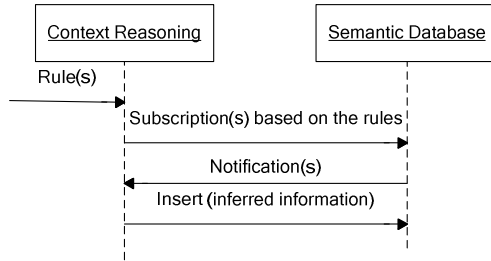


Fig. 4. Example sequence diagram of the context reasoning agent

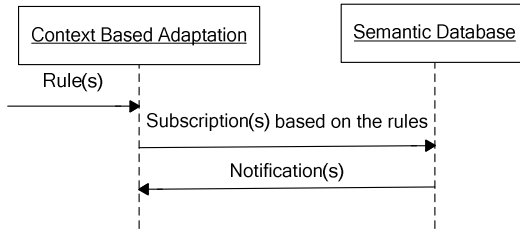


Fig. 5. Example sequence diagram of the context-based adaptation agent

## 4 Validation

We validated the context-awareness micro-architecture with two scenarios: 1) a wake-up of a person according to the first meeting of the current day (if it is a workday) and the time usually used for the morning activities after wake-up and 2) making coffee ready when the person is woke up. The first scenario is located in the personal smart space and combined with the person’s smart home (i.e. the second scenario). In the IOP the information is usually utilized from one SIB (a term for the semantic database in the IOP) but in this scenario we have a cross-domain aspect that utilizes the context information from two SS - the personal smart space and smart home; as shown in Fig. 6.

We designed and generated code for our scenarios by the Smart Modeller tool, the tool intended for end-users modeling. The idea of the Smart Modeller is to enable non-programmers to develop smart environment applications without prior knowledge on programming [16]. The Smart Modeller is an Eclipse plug-in and; more



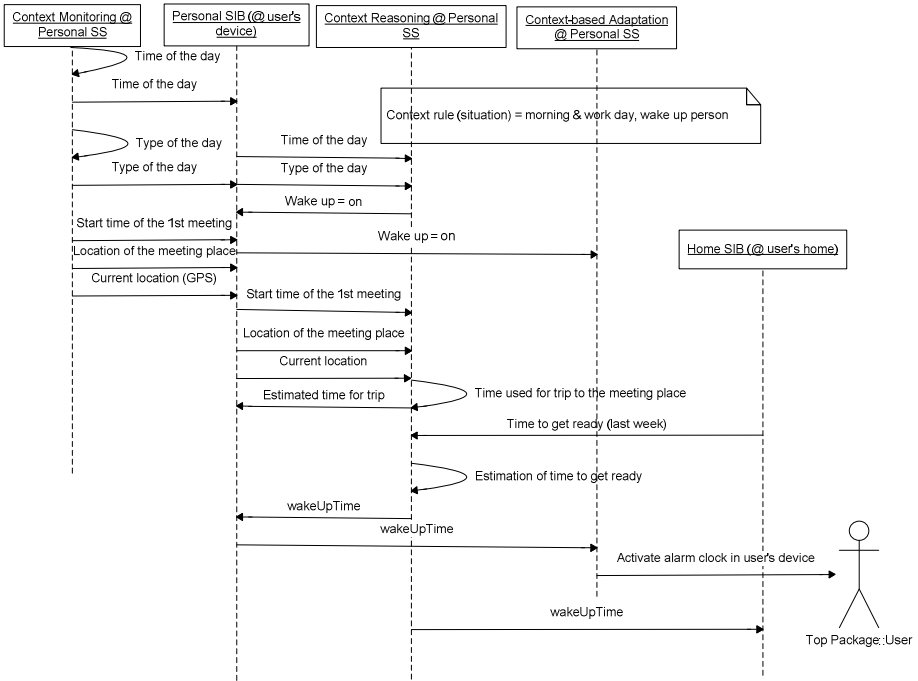


Fig. 6. Cross-domain scenario to wake up the person in the correct time

details are available in [16, 17]. Our aim was to validate the creation of the agents (KPs) based on the context-awareness micro-architecture with the design-time configuration. We configured the rules and ontology with the tool. Consequently, we decided not to save the rules to the SIB as text strings. We generated the code and started the software again when we wanted to update the rules. Fig. 6 illustrates the main activities of each agent and the flow of the information between agents and SIBs. The context monitoring agent inserts the information and the context-based adaptation agent receives the notifications. We left out the subscriptions to the SIB to keep the drawing readable. The context reasoning agent communicates with both the personal SIB and home SIB by inserting and receiving the information. Based on the wake-up time inserted to the home SIB by the context reasoning agent from the personal space, the context-awareness agents in the smart home are able to prepare the needed monitoring, reasoning and adaption actions so that the coffee will be ready at wake-up time. We managed to wake up the person in the right time and switch on the coffee maker as was targeted, even though the coffee maker was a virtual one at this time. Thus, we were able to reason on real world situations and adapt the behavior of the smart space application accordingly. We utilized two times the context-awareness micro-architecture and based on that we can claim that our micro-architecture fulfilled its purpose.

We learnt that we need to further develop our context ontology to enable its expansion with the domain specific ontologies. We discovered that it is challenging to create convenient, general and relevant concepts for the context ontology that is suitable for

the heterogeneous smart spaces. Hence, with the enhanced context ontology we can better handle the heterogeneity of context information. With the help of dynamically updatable rules we could also change the execution place of a single agent easily. For example, monitoring and reasoning agents could be activated on another computer instead of the user's mobile device in a case where battery runs low or more processing power is needed. In summary, the more complex the real world situation is, the more challenging it is to define the rules for that situation.

## 5 Conclusions

In this paper, we introduced the novel context-awareness micro-architecture to design software in a way that it is able to react to changes based on information gathered via the context monitoring agent and inferred by the context reasoning agent. We validated the context-awareness micro-architecture by implementing two scenarios. In the first scenario the context reasoning agent communicated with the two semantic databases (SIBs). The validation showed that the context-awareness micro-architecture provides agents for building up scalable and interoperable smart space applications. The cross-domain scenario, the wake-up, showed that different smart spaces can be smoothly connected by the agent (KP) that is capable of communicating with multiple smart spaces. This capability is due to the context ontology used in both the smart spaces. Thus, the usage of the ontologies enables smooth evolution for the smart space applications. We see that the run-time approach will be the most beneficial approach to instantiate the context-awareness micro-architecture because it enables to update the smart space application at run-time with new rules and agents. The side effect for smooth and dynamic updating is the laborious development work.

Even though we only validated the design-time approach, we became convinced of the context-awareness micro-architecture and its usefulness. Hence, with the context-awareness micro-architecture we were able to tackle the above mentioned challenges: 1) the heterogeneity of context information handling, 2) reasoning on real world situations, and 3) doing adaptations based on the context information. In the future, we will concentrate on 1) enhancing the context ontology, and 2) providing the needed rules at run-time.

## Acknowledgements

This work has been funded by Tekes, VTT, and the European Commission, in the framework of the ARTEMIS JU SP3 SOFIA project.

## References

1. Hong, J., Suh, E., Kim, S.: Context-aware systems: A literature review and classification. *Expert System with Applications* 36(4), 8502–8509 (2009)
2. Bettini, C., Brdiczka, O., Henriksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* 6(2), 161–180 (2010)

3. Truong, H., Dustdar, S.: A Survey on Context-aware Web Service Systems. *International Journal of Web Information Systems* 5(1), 5–31 (2009)
4. Indulska, J., Nicklas, D.: Introduction to the special issue on context modelling, reasoning and management. *Pervasive and Mobile Computing* 6(2), 159–160 (2010)
5. Guéhéneuc, Y.: P-MARt: Pattern-like Micro Architecture Repository. In: Weiss, M., Birukou, A., Giorgini, P. (eds.) *Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories* (2007)
6. Laddaga, R.: Active Software. In: Robertson, P., Shrobe, H.E., Laddaga, R. (eds.) *IWSAS 2000. LNCS, vol. 1936*, pp. 11–26. Springer, Heidelberg (2001)
7. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing (1999)
8. Eugster, P.T., Garbinato, B., Holzer, A.: Middleware Support for Context-aware Applications. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) *Middleware for Network Eccentric and Mobile Applications*, pp. 305–322. Springer, Heidelberg (2009)
9. SOFIA IOP, <http://www.sofia-project.eu/>
10. Smart-M3, <http://sourceforge.net/projects/smart-m3/>
11. Toninelli, A., Pantsar-Syväniemi, S., Bellavista, P., Ovaska, E.: Supporting Context Awareness in Smart Environments: a Scalable Approach to Information Interoperability. In: *M-MPAC 2010*, session: short papers, article no: 5. ACM, IFIP, USENIX (2009)
12. Pantsar-Syväniemi, S., Simula, K., Ovaska, E.: Context-awareness in Smart Spaces. In: *SISS 2010*, pp. 1023–1028. IEEE Press, New York (2010)
13. Chen, H., Finin, T., Joshi, A.: *The SOUPA Ontology for Pervasive Computing*. Whitestein Series in Software Agent Technologies. Springer, Heidelberg (2005)
14. Soyly, A., De Causmaecker, P., Desmet, P.: Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering. *Journal of Software* 4(9), 992–1013 (2009)
15. Evesti, A., Pantsar-Syväniemi, S.: Towards Micro Architecture for Security Adaption. In: *MeSSA 2010*, pp. 181–188. ACM, New York (2010)
16. Katasonov, A.: Enabling non-programmers to develop smart environment applications. In: *SISS 2010*, pp. 1059–1064. IEEE Press, USA (2010)
17. Katasonov, A., Palviainen, M.: Towards ontology-driven development of applications for smart environments. In: *The 8th PERCOM Workshops*, pp. 696–701. IEEE Press, USA (2010)

# Distributed Web Service Architecture for Scalable Content Analysis: Semi-automatic Annotation of User Generated Content

Mika Rautiainen, Arto Heikkinen, Jouni Sarvanko, and Mika Ylianttila

MediaTeam Research Group, University of Oulu, Department of Electrical and Information Engineering, Computer Science and Engineering Laboratory, P.O. BOX 4500  
FIN-90014 UNIVERSITY OF OULU, Finland

{mika.rautiainen, arto.heikkinen, jouni.sarvanko,  
mika.ylianttila}@ee.oulu.fi

**Abstract.** This paper describes distributed web service architecture to allow web level scalability for multimedia content analysis. We introduce service component architecture that allows for orchestration and composition of services as content analysis units to enrich multimedia metadata using common data model built upon MPEG-7 content description standard. We report experiments with automatic load balancing and introduce an end user application that facilitates annotation of people in images and video.

**Keywords:** distributed multimedia analysis, web services, user generated content, MPEG-7.

## 1 Introduction

Image and video sharing services have become one of the most popular service types in the Internet due to increasing creative use and communication through web and multimedia. The amount of available content is constantly growing with an accelerating rate. As an example, YouTube [9] has reported that every minute 20 hours of new video material is uploaded by its users. In a day, that makes 1,200 days, or roughly over three years worth of video content. It is easy to grasp that with such huge volumes, it is very important to describe and annotate the content with adequate accuracy to guarantee accessibility in variety of use scenarios.

Content-based analysis methods have been a popular research topic since the early 1990's. They can be utilized in fully automatic annotation or at least in assisting the user in the content description process. Content-based multimedia analysis is a computationally intensive process where distributed computing plays an important role. Web service technology has been one of the most notable technological solutions lately in the field of distributed computing. One of its most advertised features is that it is build on top of common standard web technologies, such as HTTP and XML.

One of the recent research topics related to distributed multimedia content analysis is using web service technologies in the distribution of analysis tasks. This research topic is very new and only a few web service based solutions exist at the moment.

Heinzl et al. [1] present a service-oriented infrastructure for multimedia applications. Their solution aims to provide tools for developing and using a variety of multimedia applications covering video and audio content analysis, audio synthesis and multimedia consumption. The described system uses the SOAP [2] protocol, Business Process Execution Language (BPEL) [3] and Flexible SOAP with Attachments (Flex-SwA) [4] for transmitting the multimedia data. Ewerth et al. [5] introduce a distributed solution for detecting cut points from video files. The presented architecture is based on grid computing paradigms and Globus toolkit and it utilizes web services and SOAP technologies in the implementation.

Other than web service based systems of distributed multimedia analysis have been more common. Seinstra et al. [6] present a distributed solution for video content analysis. They introduce Parallel-Horus, a cluster programming library that allows implementation of distributed multimedia applications as fully sequential programs. The system uses a grid computing execution model. The CASSANDRA framework [7] is a modular, self-aware, self-organizing, real-time and distributed multimedia content analysis system based on Service Oriented Architecture and dynamic resource utilization.

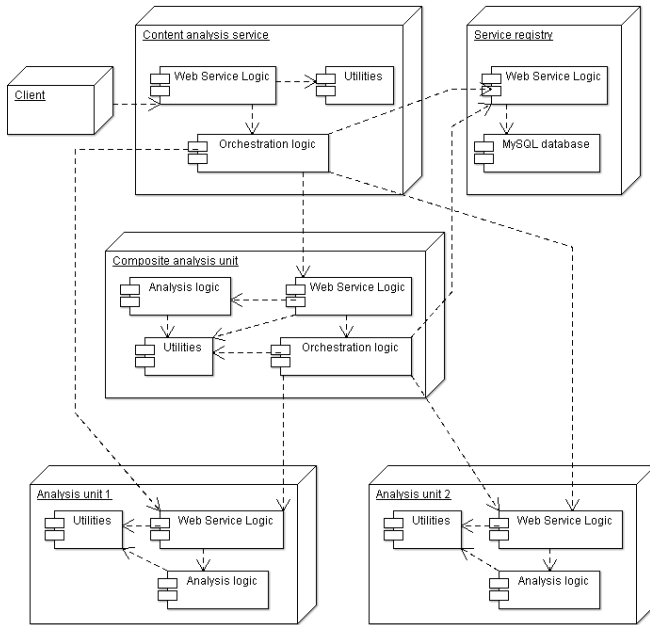
This paper presents a service platform for distributed multimedia content analysis. The platform enables versatile analysis of different content types, including video, image, audio and textual content. The paper introduces a content annotation web application for the platform and reports experiments with automatic distribution of analysis tasks. The paper is structured as follows. Section 2 describes the distributed content analysis service platform based on modular analysis units, resource aware load balancing and web service communication. Section 3 introduces content annotation web application for end users. Section 4 reports experiments with distributed content analysis and Section 5 concludes the paper.

## 2 Distributed Content Analysis Service Platform

### 2.1 Service Composition

The architecture of the multimedia content analysis platform is based on a service-oriented design, where the distributed application logic is accessible via web service interfaces. The system consists of four basic entities: Client software, Content analysis service, Analysis unit(s) and Service registry. A system diagram is shown in Fig. 1. Communication between different nodes uses XML-based SOAP [2] messages that are transported via HTTP. The service interfaces are defined using Web Service Description Language (WSDL) [3].

A client can be any software program that is able to utilize the analysis output provided by the platform. Content analysis service is the control point of the system. It discovers and invokes the analysis units required to fulfill the client's requests. It provides a web service interface to the platform clients, therefore acting as the single entry point to the platform. The analysis units are web services that execute the actual content analysis tasks. Each unit specializes to a specific analysis functionality. The service registry keeps a MySQL database containing information about registered analysis units. It also provides a web service interface for discovering and registering



**Fig. 1.** Architectural overview of the system

services. Orchestration logic is included in all services that can invoke other services. It contains tools for service discovery, selection and invocation that enable dynamically formed service compositions.

The content file must naturally be transferred to the platform in order to analyze it. The transfer of the content file is done using HTTP protocol. The file is first fetched from its original location by the content analysis service after the client has sent the analysis request. It is then distributed from the content analysis service to every analysis units participating in the analysis process.

One important feature of the platform is that a content analysis unit may utilize the results of the other analysis units to provide more high-level results. As an example, a person identification analysis unit could utilize the results from speech detection and face detection units. In this situation, the unit forms a service composition with the other units it utilizes. The composite unit performs similar operations as the content analysis service: it discovers, selects and invokes the analysis units that are part of the composition. In situations where a single analysis functionality is invoked by multiple compositions for the same data, redundant invocation is avoided by selecting the analysis unit that is already processing the data.

The platform is also capable of balancing the load between different services in a situation where an analysis unit has been replicated to several nodes. The developed method is more sophisticated than a round-robin method where requests are distributed to the different servers in a circular order. The method balances the load in a resource-aware manner, where the server's current load level and its computational capacity are taken into account. For determining the server load level, the load

average reported by unix-based operating systems is used. Based on the load level  $l$  and the number of CPU cores in the server  $N_{CPU}$  the load balancing method calculates a value representing the current available processing capacity of the server  $C$ . The orchestration logic sends a web service call to every candidate node requesting a load level of each machine. The node with the highest available processing capacity is always selected. Eq. 1 defines the formula used in the calculation.

$$C(l, N_{CPU}) = \begin{cases} N_{CPU} - l & , l \leq N_{CPU} \\ (N_{CPU} - l)/N_{CPU} & , l > N_{CPU} \end{cases} \quad (1)$$

If the load level is equal or less than the total number of CPU cores, the processing capacity is calculated by subtracting the load value from the number of CPU cores. This gives a value representing the average number of “free” CPUs in the server. In an overload situation where the load level is higher than the number of CPU cores, the subtraction results in a negative number. In this case, the subtraction result is divided by the number of CPU cores in order to reflect the relative overload of the server. Exemplary values calculated using Eq. 1 can be seen in Table 1.

**Table 1.** Available processing capacity values calculated with different load levels and number of CPUs

Load level	C, $N_{CPU} = 2$	C, $N_{CPU} = 4$	C, $N_{CPU} = 8$	C, $N_{CPU} = 16$
0.5	1.5	3.5	7.5	15.5
2	0	2	6	14
4	-1	0	4	12
8	-3	-1	0	8
16	-7	-3	-1	0

## 2.2 Semantic Content Description Model

The distributed content analysis platform aims at producing content description for multimedia data that is usable in several content rich applications, such as web sites for user generated content. For that purpose, each component of the platform attempts to enrich the description metadata so that the clients will ultimately receive rich semantic attributes describing the original multimedia. To support this process, the data model for semantic content description was designed using MPEG-7 [8]. The MPEG-7 segment decompositions were restricted into a four level hierarchy and the structure fixed so that the different services would know what type of data resided on each level of the hierarchy. The designed data model consisted of Media, Concept, Object and Element levels as seen in Fig. 2.

Media segment contains all the information, manual annotation and automatically analyzed concepts of the source media. It also determines the structure of the layers below. The next level, Concepts, holds the analysis results from each service. E.g. a detection service that locates people from images and video produces Person Concepts. The Objects are for individual instances of a Concept. E.g. a single detected

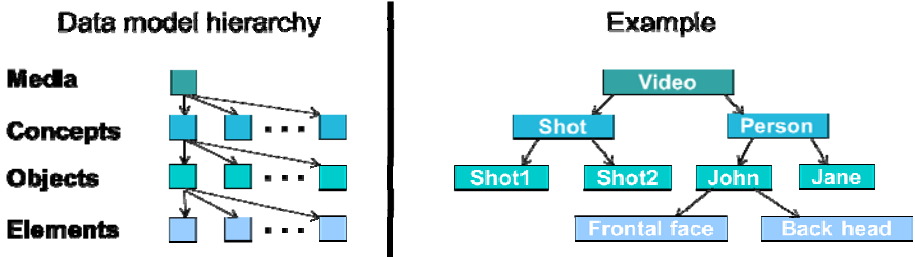


Fig. 2. Data model structure

person sequence in a video is an instance of the Person Concept. Finally, the Elements determine the physical characteristics of an instance in temporal and/or spatial domains of the media data.


### 3 Content Annotation Web Application

We developed an end user application to evaluate the feasibility of content analysis as a distributed web service. It was aimed to offer an easy way for an end user to annotate user generated videos and images on the web. To facilitate this, the application lets users to perform semi-automatic person detection based on face sequences detection with the help of the Content analysis service. The content analysis process for the user generated video was realized with a combination of services providing face detection, tracking and creation of temporal face sequences and shot segmentation for edited videos.

The web application interface is a web page, where users can upload videos and images of their choosing. After the upload phase, users can write semantic descriptions (free text and title) for the file they have uploaded and additionally use the Content analysis service to automatically process and annotate the media. Fig. 3 depicts the application interface. Upper part consists of information concerning the uploaded video and has text boxes for describing and naming the video file. Uploaded video or image can be previewed at the top of the page.

Users can request content analysis for the uploaded file, which will invoke person detection process. Once the results of automatic content analysis are ready, a result editor loads at the bottom of the page as seen in Fig. 3. Users can edit and write descriptions for the detected face sequence objects. The video is also divided into shots that are shown as thumbnails at the bottom. The face sequence objects are represented by single images. The application is designed to help users in their annotation task through simplified semantic concept summaries. In the example of Fig. 3 user sees an element that represents entire spatio-temporal (time-continuous) description of the person trajectory in a video instead of series of separate face detection samples. The application view hides the complexities of MPEG-7 metadata and system data models from the end user in order to make the user interface approachable for casual users. Once the user decides to submit the finalized metadata, an MPEG-7 file is created with all the extracted concept information.





**File name:** starwrecktest\_lyhyt.avi

**File size:** 9312084 bytes (8.9 megabytes)

**File format:** AVI

**Video format:** MPEG-4 Visual

**Video length:** 117.5 seconds ( 1 m 57.5 s )

**Video resolution:** 640x272

**Video framerate:** 25.0 fps

**Video scan type:** Progressive

**Audio format:** MPEG Audio

**\*Media content title**

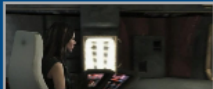
Star Wreck clip

**Textual description**


A short clip from the beginning of the movie Star Wreck

**Automatic annotation results editor**


**Shots detected from video**




Play shot (duration: 7s)



Play shot (duration: 8s)



Play shot (duration: 4s)



Play shot (duration: 3s)

**Shot annotation**


**Shot description**

Piik entering the command bridge.

**Summary of shot concepts**

Portrait composition     Several people

**Detected concept objects**



Person

Description:

James B. Piik

Cancel

Submit

**Fig. 3.** Content annotation – The web user interface

## 4 Experiments with Distributed Content Analysis

The distributed content analysis platform's ability to balance load between replicated service components on different nodes was measured with an analysis task run over a period of time. The purpose of the test was to measure how well the platform was able to balance the load between servers with different performance characteristics and a varying amount of external load.

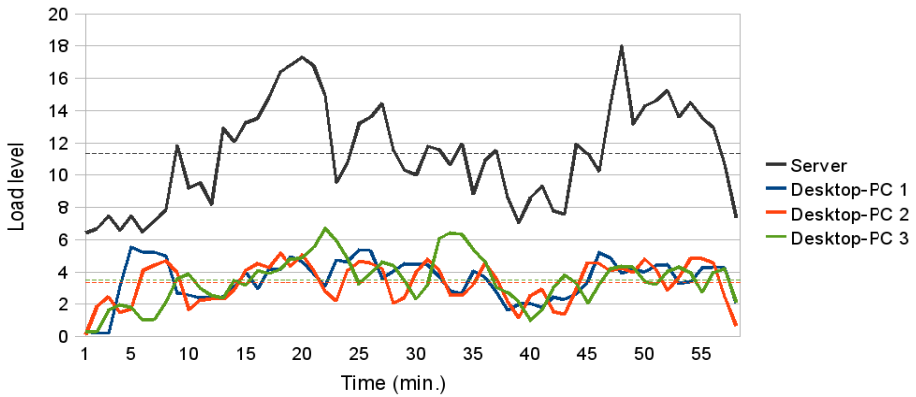
The test was performed by configuring the content analysis platform with a frontal face detection service, an analysis unit that detects frontal face regions from digital images and videos. The analysis unit was replicated into four different computers and communication within the platform was achieved through web service invocations. Frontal face detection service was selected for this test because it can efficiently utilize the entire processing capacity of the computer it runs on due to its parallelized internal design. The parallelization is particularly effective in analyzing videos.

The test environment consisted of three identical modern desktop PCs and a larger server machine. The desktop PCs were dual core machines with 2 gigabytes of memory. The server machine had two quad core CPUs totaling 8 cores and 24 gigabytes of memory. The computers were connected in the same LAN via a gigabit Ethernet connection. Apache Axis2 was used as the web service engine in the platform services that were running on Apache Tomcat servers.

The test was conducted as a one hour test run where a client application sent analysis requests to the platform requesting face detection for a video file. The requests were sent at random intervals between 15 and 45 seconds in an attempt to simulate a more realistic scenario where the analysis requests would arrive at unpredictable times. The analyzed video file was a 1 minute 35 seconds long MPEG-4/AVC video with the resolution of 640\*480 pixels captured with a mobile phone. The file size was 32 megabytes. The specifications of the selected video clip were seen to represent well the average user generated content videos.

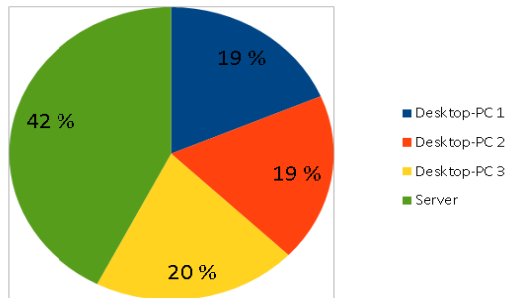
The load levels of the different machines were measured over the time period of the test run. Their development can be seen in Fig. 4. The first observation is that the curve representing the load of the server machine is at a much higher level than the curves of the desktop-PCs. The curves representing the desktop-class machines are grouped quite closely together. The dashed lines show the mean load values for each machine. The mean value for the server machine was 11.36 whereas the mean values for the desktop-PCs were very close to each other, 3.51, 3.32 and 3.48 respectively. The average load for the desktop machines was 3.44 which gives 3.3 times higher load for the server machine. Fig. 4 shows that the load levels on the server have larger variance than with the desktop computers. This is because the server had other external processes running simultaneously in the background which affected the overall load level.

The total number of analysis jobs distributed to the different nodes was also measured during the test. The proportional distribution of the analysis requests to the different machines can be seen in Fig. 5. The work was distributed equally among the desktop PCs, each receiving about 20% of the requests. The server machine processed a larger share of the requests, 42%. Therefore the server-class machine performed about 2.2 times more analysis requests than the average desktop machine. In an ideal



**Fig. 4.** Load levels on the different nodes over time

scenario the server would have processed about 4 times more requests as it has roughly 4 times more CPU power compared to a single desktop machine but simultaneous background tasks were taxing the machine. Background processes were periodically CPU-intensive data processing tasks that were assigned to the server machine, but not belonging to our service application.



**Fig. 5.** Distribution of analysis jobs to different nodes

In addition to the load balancing test, orchestration's capability to form different service compositions was tested. Four different analysis units were deployed for the test providing the following analysis functionalities: shot segmentation (SS), frontal face detection (FF), frontal face sequence detection (FFS) and visual object tracking. The frontal face sequence detection (FFS) is a composite analysis unit that utilizes the SS, FF and visual object tracking analysis units. The following compositions were tested in order to confirm the orchestration logic to be functional: SS, FF, SS+FF, FFS, FFS+SS+FF.

## 5 Discussion and Conclusions

The novelty of the developed service platform is in building a modular, distributed multimedia content analysis system using web service technologies and demonstration of its feasibility with an end user web application. Using standard web technologies results in a platform independent distributed system where service compositions can be executed on heterogeneous nodes over heterogeneous network topologies. As the messaging between the nodes is based on commonly used protocols on the Web, firewall and other networking problems are minimized. This allows a very flexible and potentially large-scale distribution of analysis tasks. All the analysis units conform to a common web service interface so they are all invoked coherently and facilitate automation and flexibility in service compositions.

The automatic load balancing was effective in distributing the tasks to dedicated desktop nodes as well as to multi-purpose processing server. In the load balancing measurement, the average load on the server machine was 3.3 times higher than on the PCs, while the optimal value would have been 4. This imperfection is a result of measuring the load level as an average value and therefore it delays the adaptation algorithm. Also the job size was fairly large and an additional job increased the load of the PCs in a higher granularity than with the server. The server received only 2.2x more analysis jobs than the PCs, whereas the ideal would have been closer to 4x. This can be partially explained with the fact that the server had other processes running in the background which affected the overall load level.

We have introduced web service based distributed content annotation platform that allows scalable content analysis and processing tasks using web service components. The components are able to receive requests from the orchestration logic through common platform interface definitions. We have also introduced an end user web application that is suitable for user generated content annotation. SOAP messaging and XML based content description metadata causes some overhead to the platform but the impact is not significant due to data intensive nature of the actual media analysis.

## Acknowledgements

We like to thank Finnish Funding Agency for Technology and Innovation (Tekes) and Academy of Finland for supporting this research.

## References

1. Heinzl, S., Seiler, D., Juhnke, E., Stadelmann, T., Ewerth, R., Grauer, M., Freisleben, B.: A scalable service-oriented architecture for multimedia analysis, synthesis and consumption. *Inderscience International Journal of Web and Grid Services* 5, 219–260 (2009)
2. W3C XML Protocol Working Group SOAP Version 1.2 specification, <http://www.w3.org/TR/soap12>
3. Christensen, E., Curbera, F., Meredith, G., Weerawana, S.: Web Services Description Language (WSDL) 1.1. W3C Note, <http://www.w3.org/TR/wsdl>

4. Heinzl, S., Mathes, M., Friese, T., Smith, M., Freisleben, B.: Flex-SwA: flexible exchange of binary data based on SOAP messages with attachments. In: Proceedings of the IEEE International Conference on Web Services, pp. 3–10 (2006)
5. Ewerth, R., Friese, T., Grube, M., Freisleben, B.: Grid services for distributed video cut detection. In: Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering, pp. 164–168 (2004)
6. Seinstra, F., Geusebroek, J., Koelma, D., Snoek, C., Worring, M., Smeulders, A.: High-performance distributed video content analysis with parallel-horus. *IEEE Multimedia* 14(4), 64–75 (2007)
7. de Lange, F., Nesvadba, J., Lukkien, J.: CASSANDRA Framework: A Service Oriented Distributed Multimedia Content Analysis Engine. In: IEEE Eight International Workshop On Image Analysis for Multimedia Interactive Services (WIAMIS 2007) (2007)
8. Manjunath, B.S., Salembier, P., Sikora, T.: Introduction to MPEG-7: Multimedia content description standard. Wiley, New York (2001)
9. YouTube – Broadcast Yourself, <http://www.youtube.com>

# MashReduce – Server-Side Mashups for Mobile Devices

Joonas Salo<sup>1</sup>, Timo Aaltonen<sup>2</sup>, and Tommi Mikkonen<sup>1</sup>

<sup>1</sup> Department of Software Systems  
Tampere University of Technology  
PL 553, 33101 Tampere, Finland  
{joonas.salo,tommi.mikkonen}@tut.fi

<sup>2</sup> Nokia Research Center Tampere  
Visiokatu 1 33720 Tampere, Finland  
timo.ta.aaltonen@nokia.com

**Abstract.** In the past few years, the Web has become a popular deployment environment for new kinds of software applications. In the new era of web-based software, applications live on the Web as services, and consist of data, code and other resources that can be located anywhere in the world. Furthermore, web sites, commonly referred to as mashups, that combine (“mash up”) content from more than one source have become common. So far, mashups have been usually built such that the client downloads and combines data. While this enables using already existing resources in new contexts, this is also a fundamental restriction. In this paper, we introduce an architecture that enables a client system to upload code to servers that then run the code to create application specific content. In addition, we demonstrate the capabilities of the system with a simple application that performs face detection for various photographs residing in the servers.

## 1 Introduction

The widespread adoption of the World Wide Web has fundamentally changed the landscape of software development. In the past few years, the Web has become a popular deployment environment for new types of software systems and applications. In the new era of web-based software, applications live on the Web as services. They consist of data, code and other resources that can be located anywhere in the world. Furthermore, web sites, commonly referred to as mashups [12,7], that combine (“mash up”) content from more than one source have become common. Mashups are content aggregates that leverage the power of the Web to support instant, worldwide sharing of content. Typical examples of mashups are web sites that combine photographs or maps taken from one site with other data (e.g., news, blog entries, weather or traffic information, or price comparison data) that are overlaid on top of the map or photo.

So far, mashups have most commonly been built such that the client downloads and combines (or mashes up) data. While this enables using already existing resources in new contexts, this is also a fundamental restriction – it is not

possible to use computing power of web servers, but only the client can perform computations. This restriction becomes obvious when aiming at applications where accessing resources requires extensive preprocessing. Moreover, when considering systems with restricted capacity, such as mobile devices or embedded systems in general, the problem becomes even more apparent [2].

In this paper, we introduce an architecture that enables a client system to upload code to servers that then execute the code to create application specific content. The approach is based on Google’s MapReduce [5] programming model for processing large data sets. In MapReduce the input set of the computation is split into reasonably sized partitions, and each item of the input is associated with a  $(key, value)$  pair. The actual computation consists of two steps:

- Map step: A map function is associated with each item of the input. The function takes a  $(key, value)$  pair as input, and maps it to a list of  $(key, value)$  pairs in another domain  $(k_{output}, v_{intermediate})$ .
- Reduce step: The MapReduce infrastructure combines the values  $v_{intermediate}$  with the same key  $k_{output}$  to pairs  $(k_{output}, list(v_{intermediate}))$ . Finally, a reduce function is executed for each of these pairs to produce a  $list(v_{output})$  for each key  $k_{output}$ . The list often contains only one output value.

This scheme allows us to sort the input data in accordance to some interesting information by using the intermediate keys between mapping and reducing phases. The key-value interface between the mapping and reducing phases is also flexible. Consequently general-purpose map and reduce functions can be combined to create meaningful applications.

To demonstrate the strengths of the proposed approach, we introduce a simple application that performs face detection for various photographs residing in the servers built using the above primitives.

The rest of this paper is structured as follows. Section 2 defines our approach for mashup development at the conceptual level. Then, Section 3 discusses our proof-of-concept implementation of the architecture, and Section 4 introduces the demo application. Section 5 discusses some directions for future work, and Section 6 draws some final conclusions.

## 2 Approach

The current trend for creating mashups is based on using JavaScript clients. They can download content from multiple sources, and render aggregated views based on the downloaded content. This approach is reasonable, when the clients are running in an environment where available resources are plenty. For example, a web browser running in a desktop computer has no difficulties in performing such computations, but in contrast a mobile device might run slowly and drain its battery rapidly under heavy processing load. For a setting with limited computational resources, we propose a model that transmits the computational load to servers to minimize clients’ processing tasks. In our model, the server accesses

and downloads the content, mashes it up, prepares a device-optimized view, and transmits the view to the device.

## 2.1 MapReduce

Our interest in MapReduce was the programming model, which allows tasks to be distributed and merged automatically behind the scenes without intervention from the programmer. The programmer needs to simply write two programs (map and reduce) that have a standardized way of handling input and output, while the MapReduce framework handles all the distribution logic. Distribution of tasks is essential for us, since the content to mashup is usually spread across different servers. In addition MapReduce adds a further twist between mapping and reducing phases, because it arranges the intermediate values by keys that provide the interesting information for us. For example, content that is constituted by images can be arranged in accordance to their location by the mappers and then one reducer is started for each set of images in a location. One more advantage from MapReduce is the possibility to combine general-purpose map and reduce functions by relying on the standardized key-value interface.

Google designed MapReduce for large-scale computations that may take minutes to start up and hours or days to complete. Web mashups, on the other hand, have strict response time requirements, because they are directly used by human users, and therefore mashup systems intended for such context should have a running time of only few seconds, if even that. Therefore, we decided to implement a lightweight version of MapReduce that can start up quickly. Our approach also contains the idea of sending the computation directly (or close) to the content by running our server software at the content server or in the same LAN. However, if this can not be arranged, we can download content to a centralized grid and process it there.

## 2.2 Conceptual Model

Our mashup system consists of one *mashup server*, whose task is to orchestrate a set of *map* and *reduce agents*. Agents are used to dig and prepare the content items for the mashup server. Individual map functions return (*key, value*) pairs, which are consumed by the associated reduce functions. Similarly, reduce functions return further (*key, value*) pairs, which are then consumed by the mashup server to create the mashup itself. The return type of the mashup server depends on the request. It can be for example a web page that is downloaded as HTML [9], detailed descriptions of items in machine-readable form like ATOM [8] feeds, or plain simple data structures, such as JSON [4].

The model is illustrated in Figure 1. The client on the top has requested a mashup, which is created by the server using two (*map, reduce*) pairs. The map functions have been delivered to be computed close to the content. Each content item is fed one-by-one to the map function. The result of computation is an ordered set of (*key, list(value)*) pairs, where MashReduce [11] infrastructure has merged the values with the same key into the list. Map functions have no



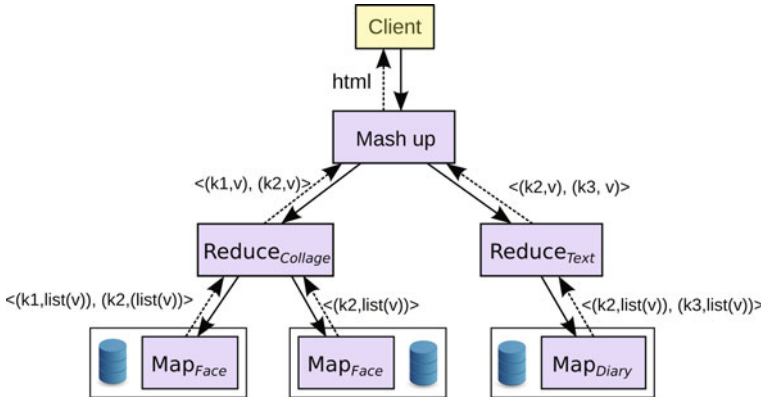


Fig. 1. Yet another three-layered architecture

side-effects to the content, but they can prepare mashup specific items from the content to some volatile memory. Reduce functions perform mashup dependent reducing operations to the set returned by the map functions, and return a set of similar type to the mashup function. Based on this data, the mashup server finally creates the actual mashup, and delivers it to the client.

As an example, consider the following case. Let us assume that the user wants to render faces of her friends on a map based on the geo-location of the image. A *(map, reduce)* pair would consist of a map function, say *Map<sub>Face</sub>*, which, based on image's URI and content as input, makes face recognition, crops a face of a friend from the image to a temporary file if such face is found, and returns pair  $(loc, url)$ , where *loc* is the geo-location and *url* refers to the cropped face.

Next, the MashReduce infrastructure delivers the output of the map function to the reduce function, sorted by *loc* in the form  $(loc, list(url))$ . The reduce function, say *Reduce<sub>Collage</sub>* takes these as input, and makes a collage image of the images from the list, since they are taken in the same location, and returns  $(loc, url)$  pair, where *loc* is the geo-location, and *url* refers to the collage. If there is only one face taken in some location, the collage need not be created, but the  $(loc, url)$  pair from *Map<sub>Face</sub>* function can be returned as such.

The mashup server first downloads a map (chart) from some map server. Then it takes the sets of the  $(loc, url)$  pairs as input, and renders the collages referred by each *url* to the corresponding location on the map. Finally, it returns result of this computation to the client.

If the user wants to render more content onto the map, then more map and reduce agents are needed. Similarly to faces of friends, a *(map, reduce)* pair could be used for digging user's diary with geo-location. Map function *Map<sub>Diary</sub>* would dig the titles and return a set of  $(loc, title)$  pairs to be reduced. Again, if two notes were made in the same geo-location, reduce function *Reduce<sub>Title</sub>* would merge them. Finally, the mashup function would again render the (possibly merged) titles onto the map.

### 2.3 Conceptual Architecture

The proposed model can be implemented with the conceptual architecture depicted in Figure 2. The client on the top is willing to receive a mashup made of content items in the two content servers in the bottom. The concepts defined above are distributed to the architectural components. Map functions are computed close to the content – in the content servers, if possible – and the mash-up function itself in the master server.

The distribution of reduce functions is more complex. They can be computed by either content or master server, or even by an additional layer of servers that could be added in between them. Moreover, some computational tasks are such that the reduction can be carried out in several phases. As an example, a content server could compute an initial round of the reduction, and these partial results could then be delivered to another server to be further reduced. For instance, the reduce function for merging the titles discussed above could be executed in several phases.

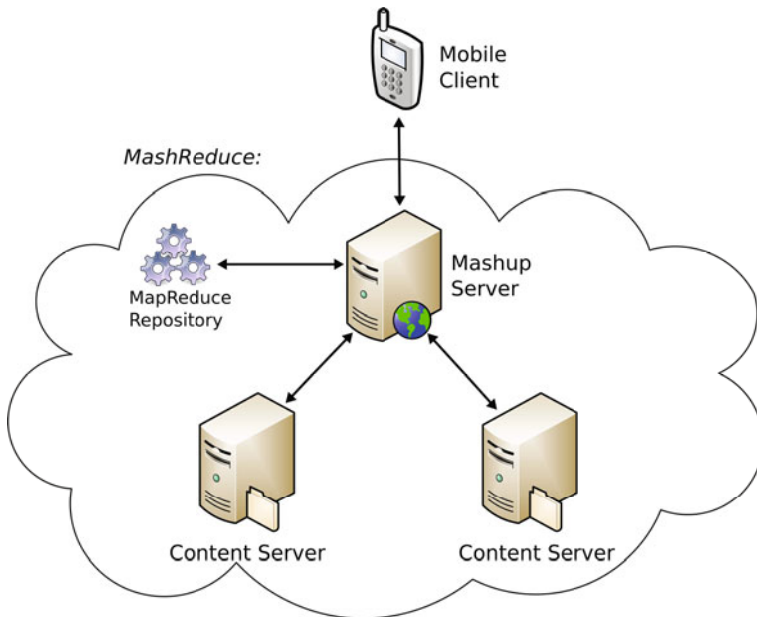


Fig. 2. Conceptual model

MapReduce repository, on the upper left-hand side in Figure 2, is a repository for universal MapReduce pairs, from which agents can be used. In addition, it is also possible to use application-specific agents that are created by the client.

### 3 Proof of Concept Implementation

The implementation of these ideas is presently work-in-progress, and they are included in a software system referred to as MashReduce. The MashReduce system consists of two main components, the mashup and the content server. Both of these are web services with an API and user interface that have been designed with the RESTful guidelines [10,6] in mind. Both map and reduce related functions have been implemented in small programs called agents. Map agents are run in the content servers and reduce agents are run in the mashup server.

By design, agents are extremely simple and there are only few rules that agent developers need to follow. The interface between the agents and the content servers consists of system level functionality such as reading files and directories and using command line parameters. This allows for very flexible implementation of the agents. As the agents do not need to link against any given libraries they can be programmed in many languages such as C, C++ and Python. Agents can be composed of arbitrary code so they can contain almost any kind of functionality, and all computing, storage, and network resources of the server device are freely available to the agent.

The MashReduce system defines two kinds of tasks, mashup tasks and map tasks. Mashup tasks are managed by the mashup server, and they contain all the details needed for creating a mashup, including the map and reduce agents as well as a listing of map tasks. Map tasks can be thought of as instances of map agents. They define the input for the agent, hold the output produced by the agent, and can in addition define the content server where the mapping is done. The mashup server is responsible for transferring the map agents to the content servers, giving the order to start the map tasks, monitoring the status of the tasks, and finally downloading the tasks output for the reduce agent. The reduce agent gets all the output from the map agents and produces the final output for the client from those files. Adhering to the principle of MapReduce, this would mean that the map functions map all values to a single key. The data that is moved between the map and reduce agents, as well as the output of the reduce agent consists of files and directories.

Ideally, the content servers would be installed close the content, meaning that they would use the same hard drive or local network. In real-life situations, however, is not always possible to get that close to the content, or the content may not even be known beforehand. The content servers provide prefetch functionality to download content for the map agents before they are started. If the content server has a good network connection it makes sense to use it for downloading the content as well as processing it. Figure 3 illustrates the prefetch process.

The map task contains enough information to express the content server to use, and what kind of content to download. MashReduce is not using any kind of distributed filesystem, but it simply downloads (or copies) files for the map agent to use as input each time it is started. If the data is available in the same hard-drive, the startup can be very fast.

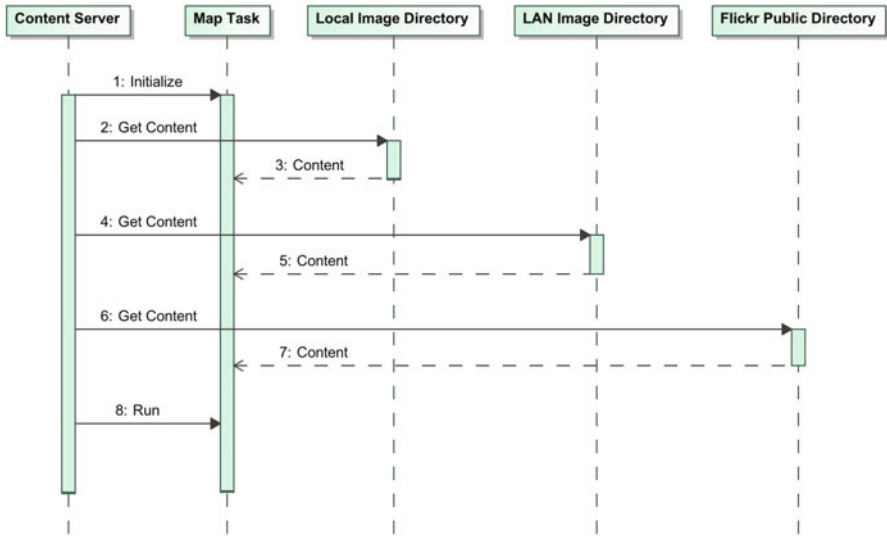


Fig. 3. Prefetching content for a map task

## 4 Face-Detection Demo

The demonstration application we have composed consists of two content servers, one mashup server and a client. The mashup server acts as a mapreduce agent-repository and provides two essential agents for this task. The map agent gets a directory of images as input, reads each image and crops each face that it finds to the output. The agent is programmed in C++ and it uses the face detection algorithm from the OpenCV [3] library. The reduce agent reads the output of all the map tasks and creates a face collage picture. It is a Python script that uses Python Image Library in its implementation.

The mashup server propagates the map task to the content servers and waits for their completion. Once it notices that all the map tasks are ready, it downloads their output and performs the reduce operation on the files. The client waits until the mashup task is ready, and then it downloads the output. Figure 4 illustrates the sequence of the demonstration. In the following, we also explain the main steps of the execution.

- The client creates a mashup task that defines the map (face detection) and reduce (image-collage) agents to use as well as the map tasks to run in the content servers.
- The mashup server propagates the map agent to the content servers and starts map tasks defined by the client.
- When the mashup server notices that all the map tasks are ready it downloads their output and runs the reduce agent. The reduce agent produces the final output (image-collage of the faces) for the client.
- The client waits until mashup task is ready and downloads the output from the mashup server.

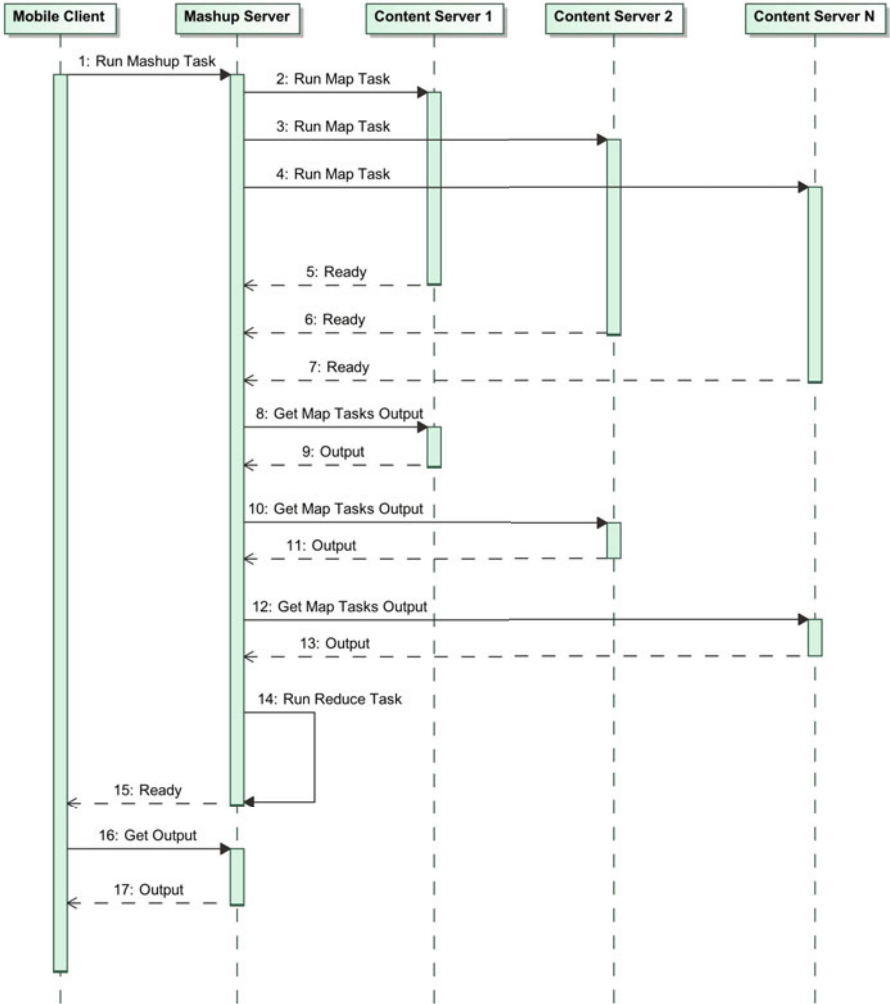


Fig. 4. Creating a mashup

## 5 Future Work

Future work for this project consist of generic architectural development as well as smaller, case-specific optimizations. The architecture will be developed towards a more structured solution that provides a clear programming model. After these steps, we look forward to building some large-scale, massive demonstrations.

MashReduce could benefit from using the MapReduce programming model in a more elaborate fashion, which would involve distributed map and reduce functions and the use of key-value pairs as form of communication.

The key-value pairs would then be used as structured communication and distribution means, but the real content would reside in the input and output files of tasks. The key-value interface between the map and agent could connect universal map and reduce agents together to form many interesting combinations. The programming model could be introduced in the form of a base class or library for agent developers.

In practice, the values could often be delivered as URIs pointing to files in tasks' outputs. The map agent could, for example, produce many output files and create key value pairs that point to these files. The framework would then download the files for the corresponding reduce agent before starting it, or the reduce agent could explicitly ask for the framework to download the files.

The reduce agents should be distributed to content servers, or some other computing servers, to not drive the mashup server to being a bottleneck for the performance. In some situations exactly the reduce agents do most of the real work while the map agents only sort the input sets in some interesting way.

Between the reducer and the client should be a modular mashup layer that transforms the output of the heavy computations into something useful for the Web environment. Some practical function for it would be to provide different representations of the mashup task such as JSON data structures, ATOM feeds or even complete Web pages. Some of the functions could be provided with general purpose mashups, but some would need to be developed especially for certain applications. The mashup layer should have some status information or even partial results from the MashReduce system that it could provide as soon as possible.

Performance and responsiveness are key values for a mashup-service and optimizing fractions of seconds in small scale can lead to saving multiple seconds as the computations grow more massive. Important steps have already been taken by replacing HTTP polling with a server-push mechanism, and precompiling C/C++ agents in the content servers before they are needed for tasks, but there still remains room tweaking the performance. Data-transfers, for example, could be enhanced by using a packaging method to return many related files in one HTTP request.

After optimizing the performance of running one task it would be time to study performance for large scale multi-user applications. A natural solution to keep constant response times with changing load levels would be to use an autoscaling cloud system, such as Amazons EC2 [\[1\]](#), that can provide a lot of raw processing and network resources. This kind of central grid complements the idea of having distributed content servers at strategic locations, and could provide a stable basis for a reliable and responsive mashup service.

## 6 Discussion

To summarize the contributions of the paper, we now have the potential of a grid computing system behind a simple RESTful interface, which we can make easily approachable with web protocols. Especially mobile devices can benefit

from this potential that enables completely new kinds of mashup applications. The use of MapReduce programming model has the advantages of making massive scale applications efficient by parallelizing computations while still keeping the development process comprehensive. After some more development on the MashReduce system we are looking forward into massive scale mashup demonstrations.

Some obstacles can be expected with licensing conditions of public web services. Many content owners wish to control how their content is shown to the user and for instance include some ads or company logos in the process. In our tests we have used Flickr's services as its licensing conditions are well-suited for our experimental purposes. In the end it would be most beneficial to actually get inside the content providers network and setup a content server where the content is actually located. This however might require radical changes in the business models of content holders, or in the fashion content is being hosted.

## References

1. Amazon Elastic Computing Cloud Homepage, <http://aws.amazon.com/ec2/> (visited: January 16, 2011)
2. Salminen, A., Nyrhinen, F., Mikkonen, T., Taivalsaari, A.: Developing client-side mashups: Experiences, guidelines and the road ahead. ACM. MindTrek (2010)
3. Bradski, G., Kaehler, A.: Learning OpenCV. O'REILLY, Sebastopol (2008)
4. Crockford, D.: RFC 4627 JSON (2006), <http://www.ietf.org/rfc/rfc4627.txt>
5. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. OSDI, p. 13 (2004)
6. Fielding, R.T.: REST: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine (2000), <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
7. Merrill, D.: Mashups: The new breed of web app. IBM Developer Works (2009)
8. Nottingham, M., Sayre, S.: RFC 4287 The Atom Syndication Format (2005), <http://tools.ietf.org/html/rfc4287>
9. Raggett, D., Hors, A.L., Jacobs, I.: HTML 4.01 Specification. W3C Recommendation (December 1999), <http://www.w3.org/TR/html4>
10. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly, Beijing (2007)
11. Salo, J.: Designing a RESTful Grid Computing System. Master's thesis, Tampere University of Technology (2010)
12. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding mashup development. IEEE Internet Computing 12, 44–52 (2008)

# Open Service Platform for Pervasive Multimedia Services Development

Juho Perälä, Daniel Pakkala, and Juhani Laitakari

VTT Technical Research Centre of Finland  
P.O. Box 1100, FI-90571 Oulu, Finland

{Juho.Perala,Daniel.Pakkala,Juhani.Laitakari}@vtt.fi

**Abstract.** The transition from delivery network and terminal oriented development of multimedia services towards multi-network and multi-terminal services is an emerging trend. This change in multimedia service development is enabled by still ongoing IP convergence of communication networks and digitalization of content. This paper presents a metadata-enabled service platform supporting development of pervasive multimedia services spanning over multiple networks and terminals, namely CAM4Home Open Platform developed in ITEA2-CAM4Home research project. The platform provides enabling services for interworking development of personalized and context aware multimedia services operating over multiple networks and multiple terminals. The platform provides a service oriented architecture and enabling services that can be used in compositions to build different types of full-featured end-to-end multimedia services. In addition, the platform introduces a novel meta-model and related metadata services enabling unified way of describing information about multimedia content, services, and user- and environment context. The platform relies on use of standardized Web Services and RDF technologies enabling operating system and network independent access to the services; thus allowing creation of pervasive and platform-independent services for any Internet connected platforms from embedded devices to mobile phones and full-featured desktop clients.

**Keywords:** multimedia, context-awareness, service platform and service development.

## 1 Introduction

Traditionally multimedia services have been developed for isolated specialized multimedia systems with specific network and terminal configuration in mind. Enabled by the still ongoing trends of content digitalization and IP convergence [1] of communication networks, transition from delivery network and terminal oriented development of multimedia services towards multi-network and multi-terminal services is an emerging trend. This transition leads to the original research problem behind the CAM4Home Open Platform: *how to support development of multi-terminal and multi-network context-aware and personalized multimedia services?*

Aim of the ITEA2-CAM4Home (Collaborative Aggregated Multimedia for Digital Home) [2] research project was to create a metadata enabled content delivery



framework enabling end-users and commercial content providers to create and deliver rich multimedia experiences. The main focus of the project was to create framework supporting concept of collaborative aggregated multimedia (CAM) that refers to aggregation and composition of individual multimedia contents into a content bundles that may include references to content-based services and can be delivered to end-users over various communication channels [3].

The CAM4Home Open Platform presented in this paper provides a solution for development and deployment of multimedia services that are interoperable with relevant multimedia- and context-metadata standards and content representation technologies in multi-network and multi-terminal computing infrastructure.

The structure of the paper is as follows. Section 2 presents the requirements set for the envisioned platform and the design approach taken based on the identified requirements. Section 3 provides detailed presentation of the architecture of the developed service platform and accompanied elementary services. Section 4 presents validation of the platform via example service developed on top of the platform. Finally, Section 5 concludes the paper with conclusion.

## 2 Requirements and Design Approach

The main goal of the platform was to support development and integration of multimedia applications and services that were developed within the project consortium. The platform development was started by defining and analyzing relevant use cases and scenarios to be supported by the platform, and by identifying all enabling technologies related to these scenarios. As a result of this analysis 13 multimedia-oriented scenarios, with over 100 individual use cases, were defined. Approximately 50 different enabling technologies (programming languages, operating systems, communication protocols, etc.) were identified that were relevant for the defined scenarios. Based on the scenario analysis, two high-level requirements were set for the developed platform.

First, the overall platform must provide meta-model and metadata framework, earlier presented in [3], supporting unified description of multimedia contents (audio, video, pictures, applications, documents, etc.) and their semantic relation and aggregation information. In addition of content information, the metadata model must support description of user, device and network profiles to facilitate context-aware content adaptation, delivery and recommendations. The metadata model must also support interoperability between existing and legacy content and context metadata formats (e.g., MPEG-7 and FOAF), and it shall be extendable to support future multimedia formats and media types.

Second, the platform must provide device- and hardware-independent service platform supporting dynamic management of services and applications in the platform. The service platform must be able to support dynamic registration and removal of services from the platform, and provide access control, discovery and binding mechanisms to support seamless service discovery and access. In addition, the service platform must provide elementary services enabling easy development and integration of applications supporting the identified use cases and scenarios.

The scenario analysis confirmed assumption that no single monolithic system could be able to support all identified scenarios and technologies. Thereby the

CAM4Home Open Platform design follows the principles and concepts of the service-oriented architecture (SOA) [4], where networked resources are exposed through their well-defined service interfaces, enabling access to these service resources without knowledge of the underlying implementation decisions. This enabled encapsulation of underlying implementation technologies and unifying access to different resources in harmonized way.

The actual service platform design work was carried out by applying principles of model-driven architectures (MDA) methodology [5] where the platform is first modeled in technology-independent manner using UML modeling language. The MDA approach enables design of the service platform and its services without dependencies to the implementation technologies, thus, the designed platform can be easily realized in virtually any service-oriented technologies. The selected MDA approach provides also future-proof designs which can be easily re-implemented on future technologies.

After technology-independent design was finalized the implementation of the service platform was realized using standardized Web Services technologies. The technology-independent service designs were transformed into Web Service Description Language (WSDL) [6] service interfaces, which were used to generate service implementations supporting the Simple Object Access Protocol (SOAP) [7] communications.

### 3 Service Platform and Architecture

Based on the identified scenarios the common functionalities were identified and set of elementary services supporting these functionalities was defined. In service identification and categorization process the identified services were grouped into two service categories: Core Platform Services and Metadata Services. The overview of the service platform and identified elementary services are presented in Figure 1.

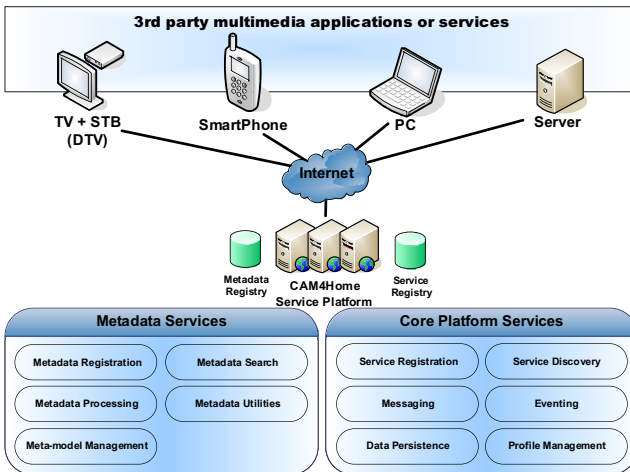
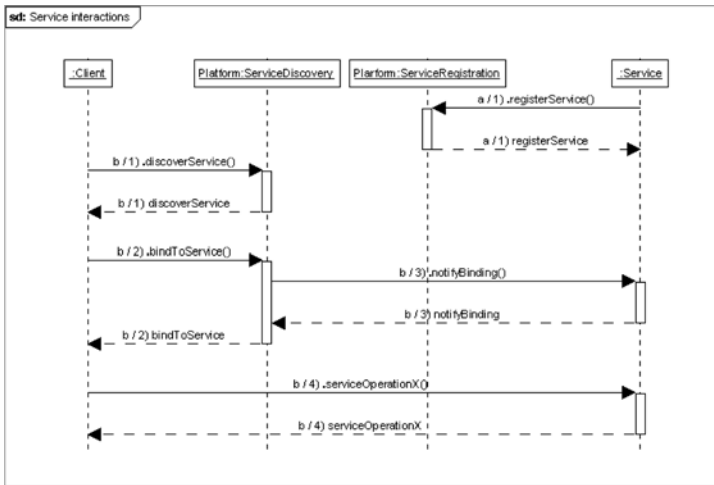


Fig. 1. CAM4Home Open Platform Service Architecture

The platform provides elementary services and necessary service and metadata registries to enable development of 3<sup>rd</sup> party applications and services. As Web-based platform, based on standardized technologies, the applications and services can be developed on virtually any device or terminal platform supporting IP-based network connectivity.

### 3.1 Service Architecture

Each service needs to be registered to be part of service platform. In registration a service description including necessary information to discover, bind and access the service is provided. The service description provides also information of service control interface, which facilitates controlling of services by the service registry. The overall service access process, including service registration (by service), service discovery (by client) and service binding, is presented in Figure 2.



**Fig. 2.** Service interactions (registration, discovery and binding)

After successful registration the service can be discovered and bound to by other applications and services. The binding operation facilitates authenticated binding of client and service endpoints for service access. In binding the platform identifies both endpoints and shares a unique binding key between the client and service endpoints. By this the service can uniquely identify incoming service requests and allow or deny service access for each client. The binding process is always initiated by the client and managed by the service platform that is responsible for authentication of clients accessing the platform services.

### 3.2 Core Platform Services

The Core Platform Services provide all platform functionalities related to management of the services and their information. These include registration of services into

the platform, binding to other services, management of service data, and management of user, device and network profile information. Detailed information of Core Platform Services is presented in following.

**Service Registration and Discovery.** Service Registration service provides means to handle dynamic registration of new services into the service registry so that they can be searched and used by other services or applications. Similarly, the Service Discovery service provides means to search and utilize services that are available in the platform. Each service is registered in the platform with service-specific description providing necessary information for other application and services to identify and use it. Registered services are expressed in the service registry by their service descriptions whose information is used as a base for the search query.

Service registry is also responsible for handling service authentication and access control. Service registration requests without allowed access rights are therefore rejected by the service registry. Service registry also controls applications and services access to other services. This is facilitated by the binding procedure that needs to be performed by any application to obtain access for wanted services.

**Eventing and Messaging.** Messaging service provides means for applications and services to send arbitrary type of messages to a single or to multiple targets regardless the network domain. Service can be used to send and receive messages between local (on same computational node) and remote (on remote networked node) applications and services within the platform. The messages send between the entities can be, for example, control messages to communicate with two separate applications or text based chat messages of a chatting user service.

Eventing service provides the applications and services a functionality to raise and listen arbitrary type of events within the platform. When an event is raised in the system all the entities that have registered to listen to the event type in question will receive an event notification. The raised events can be used, for example, to distribute general notifications about important state changes to other application or service, or to synchronize distributed applications within the platform.

Both Eventing and Messaging services provide similar functionality, but the nature of communication is different. Messaging service provides strictly point-to-(multi)point communications where Eventing service provides more broadcast-oriented means for communications.

**Profile Management.** Profile Management service is specifically designed to enable management of user and environment profiles by providing useful methods for accessing, manipulating, observing and searching the profile data. In addition, the service takes into account privacy issues of user profiles that might contain sensitive information which must be treated in secure ways keeping its access restricted. Due to these extensive profile management features, the service is also in charge of managing the user context model that has been presented in [8], hence enabling easy integration and full utilization of model's features with different types of recommender systems.

Profile Management service offers methods for adding and removing profiles, search profile data using a well known RDF [9] query language called SPARQL [10] and retrieving profiles using a UID defined in the profile data. For keeping sensitive information safe, it asks for credentials when the accessed profile information is private. Besides, it offers the clients a possibility to register an Eventing service listener

for certain information in the profile data which allows them to receive notifications when the information in the profile has been changed. This data change observation feature allows, for example, recommendation services to dynamically adapt their recommendations depending on the user status, location, etc.

**Data Persistence.** Data Persistence service provides means for the cross-network services and applications to use a common persistent storage for storing and retrieving data over different networks. Typical use of Data Persistence service includes, for example, storing user and application data to be available between different client terminals and application sessions. Another use of Data Persistence service is inter-service data storage, for example, temporal storing of images created of Web Camera service to be available for Image Processing service.

### 3.3 Metadata Framework Services

The Metadata Services provide all platform functionalities related to management of the multimedia content information. These include registration of new content into the platform, aggregation of contents as multimedia bundles, modification of content information, and management of metadata schema information. The Metadata Services rely on use of the CAM4Home meta-model presented in [3]. Detailed information of Metadata Services is presented in following.

**Metadata Registration and Search.** Metadata Registration service is used to register multimedia content metadata to the metadata registry. Metadata Registration provides an interface to a common database (i.e., metadata registry) which allows the multimedia content metadata to be registered to the platform and, thus, providing a way to discover and distribute the information to all network domains. Registering the multimedia metadata allows them to be found by other applications and services that are utilizing the Metadata Search service. Each metadata registration triggers a registration event which is notified to listeners via Eventing service.

Metadata Search service provides means for other applications and services to search multimedia content metadata that has been registered in the platform. The search is based on criteria that are expressed either with keywords or more complex search queries expressed via SPARQL query language. The given criteria is matched against the multimedia content metadata that is stored to the system and all content metadata fitting the search pattern or keywords are returned to the requesting service or application.

**Metadata Processing.** Metadata Processing service provides means to interpret the metadata so that it can be used and modified. Metadata Processing service acts as an interpreter of the CAM4Home meta-model meaning that it understands the metadata model and its instantiated format so that it can be processed. The service is basically able to:

- Read any partial or full instantiation of the meta-model from a file.
- Transform the file into a metadata model so that it can be processed.
- Validate metadata files against the current meta-model schema.
- Provide means to modify (add, remove, update) metadata.
- Store a metadata model back to file format when processing is done.

**Metadata Utilities.** Metadata Utilities is a utility service that provides operations for accomplishing more special kinds of tasks. Metadata Utilities service provides operations, for example, for manipulation of community created metadata (e.g., user comments and rating related to particular multimedia content) directly in the registry without triggering a version increment. Additionally, service can be used to register Eventing filters in order to receive notifications when certain kinds of content metadata have been registered. Service also provides means to search for partial content metadata in the registry.

**Meta-model Management.** The Meta-model Management service is used to dynamically extend the CAM4Home meta-model schema during run-time of the platform. The meta-model schema is used for specifying the structure for the metadata that is transported on the platform, and Meta-model Management service provides support for the extensibility features of the meta-model. Service design ideology of the CAM4Home platform services allows this kind of run-time extension of the data format as their service interfaces are not bound to the current data structure in use. Meta-model Management provides operations for adding and removing schema extension during the run-time of platform, and also retrieving the current schema (with the extensions) that is in use on the platform.

## 4 Validation

This section presents a case study where the described CAM Open Platform was used to implement a context-aware multimedia service for pervasive multi-terminal environment. The developed service enables end-users to access and share their favorite multimedia content from different terminals such PCs, tablets and mobile phones, and to get personalized content recommendations based on their context information. The goal of the trial system was to validate the CAM4Home Open Platform as an enabler of fast design, development and deployment of context-aware services for pervasive multi-platform environments.

A typical application use case is depicted in the Figure 3. In the use case two users, Alice and Bob, are identified. Alice is responsible for sharing multimedia content to other users. The content uploading and sharing can be performed either via web-based PC client or with a dedicated mobile phone client. The second user Bob is connected to the service via mobile phone and he wants to receive content matching to his personal interests. In the use case the user Bob will get automatic notification of available content based on his context information.

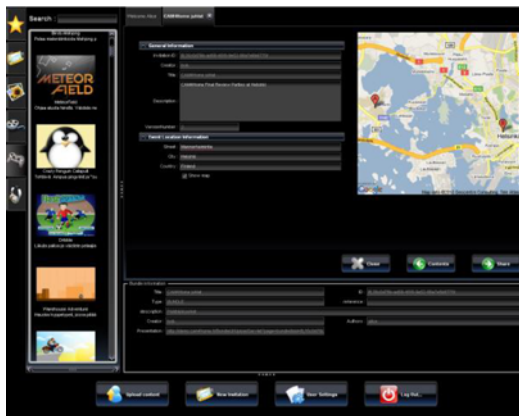


**Fig. 3.** Context-aware multimedia use case

The presented metadata framework and related metal-model was used to gather and utilize the context-information relevant for the use case. The mobile phone client application was used to automatic gathering of user context-information and it was updated to the user's device profile in the platform through Profile Management service. The automatic context-information gathering included collection user's locations information (based on phone GPS receiver), user's presence (normal, office, silent, and offline), and user's latest activities (web browsing and multimedia player history). In addition of automatic context-information retrieval from mobile devices, the users were able to manually set additional information about their presence and also provide information about their interests (e.g., hobbies, etc.). The metadata framework was also used to store and utilize the content metadata information related to the multimedia content. A recommendation service was developed to facilitate matching content metadata with the user's current context information. The recommendation service relies on use of the Metadata Search and Profile Management services to perform actual matching between the content and context information.

Another application use case was related to cross-linking the user context and multimedia content information to create personalized multimedia invitations. Multimedia invitation refers to messages shared between users that can contain references to related multimedia content, and can be linked to a related physical event and location. When a user receives a invitation, the user location information were used to personalize the invitation by showing a map and route plan from user's current location to the location associated with the invitation. An example of typical invitation is a party invitation that has associated multimedia content (e.g., music and social games for party) and that is associated with the party location. A user interface with personalized map information is presented in Figure 4. Full description of the service developed for platform validation is presented in demonstration video in [11].

As validated by the case studies, the proposed CAM4Home Open Platform provides an environment for fast development of context-aware multimedia services and applications for pervasive environments. As the service platform provides easily reusable services to support basic functionalities required for content- and context-information



**Fig. 4.** Personalized user interface based on user context-information

gathering and utilization in multi-platform environments, the platform is especially suitable for trialing and prototyping new types of multimedia- and context-based services and applications.

## 5 Conclusion

In this paper a novel service platform referred as CAM4Home Open Platform was introduced. The platform enables multi-terminal and multi-network development of context-aware pervasive multimedia services. This was achieved by identifying and providing two categories of elementary services, core platform services and metadata services, supporting the basic functionalities required for service development. A service platform such as the one presented in this paper enables fast development and prototyping of new applications and services as the common functionality provided by the service platform can be easily integrated and utilized in 3<sup>rd</sup> party applications without excessive implementation work.

As presented in the validation chapter, the presented service platform and accompanied meta-model facilitates development of multimedia services supporting context-aware use cases and scenarios based on user, device and network profiles and content information. As the service platform is deployed as server-side platform and accessed via standardized Web Service interfaces, the platform and its services can be accessed and used by any Internet-based networked device regardless of the underlying operating system or hardware platform; thereby it facilitates development of truly pervasive and platform-independent services and applications to any computing platform.

The presented platform is released as an open platform with public APIs and it can be freely used for research and non-commercial service development [11].

## Acknowledgments

This research was part of the European ITEA2-CAM4Home –project (2007-2010) funded in Finland by the National Technology Agency of Finland (TEKES).

## References

1. Decina, M., Trecordi, V.: Convergence of Telecommunications and Computing to Networking Models for Integrated Services and Applications. *Proceedings of the IEEE* 85(12), 1887–1914 (1997)
2. ITEA2-CAM4Home, <http://www.cam4home-itea2.org/>
3. Bilasco, I.M., Amir, S., Blandin, P., Djeraba, C., Laitakari, J., Martinet, J., Gracia, E.M., Pakkala, D., Rautiainen, M., Ylianttila, M., Zhou, J.: Semantics for Intelligent Delivery of Multimedia Content. In: *Proceedings of the 25th ACM Symposium on Applied Computing*, pp. 1366–1372 (March 2010)
4. Erl, T.: *SOA Principles of Service Design*. Prentice Hall, New Jersey (2007)
5. Gasevic, D., Djuric, D., Devedzic, V.: *Model Driven Architecture and Ontology Development*. Springer, Berlin (2006)



6. W3C: Web Service Definition Language (WSDL), <http://www.w3.org/TR/wsdl/>
7. W3C: Simple Object Access Protocol (SOAP), <http://www.w3.org/TR/soap/>
8. Durán, J.I., Laitakari, J., Pakkala, D., Perälä, J.: A User Meta-model for Context-Aware Recommender Systems. In: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, pp. 63–66 (September 2010)
9. W3C: Resource Description Framework (RDF), <http://www.w3.org/RDF/>
10. W3C: SPARQL Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>
11. CAM4Home Open Platform, <http://openplatform.cam4home.fi/>

# Gridification of a Radiotherapy Dose Computation Application with the XtremWeb-CH Environment

Nabil Abdennhader<sup>1</sup>, Mohamed Ben Belgacem<sup>1</sup>, Raphaël Couturier<sup>2</sup>, David Laiymani<sup>2</sup>, Sébastien Miquée<sup>2</sup>, Marko Niinimäki<sup>1</sup>, and Marc Sauget<sup>3</sup>

<sup>1</sup> University of Applied Sciences, Western Switzerland, hepia Geneva, Switzerland  
nabil.abdennadher@hesge.ch, mohamed.benbelgacem@unige.ch,  
markopekka.niinimaeki@hesge.ch

<sup>2</sup> Laboratoire d'Informatique de l'université de Franche-Comté  
IUT Belfort-Montbéliard, Rue Engel Gros, 90016 Belfort - France  
{raphael.couturier,david.laiymani,sebastien.miquee}@univ-fcomte.fr

<sup>3</sup> FEMTO-ST, ENISYS/IRMA, F-25210 Montbéliard, France  
marc.sauget@univ-fcomte.fr

**Abstract.** This paper presents the design and the evaluation of the gridification of a radiotherapy dose computation application. Due to the inherent characteristics of the application and its execution, we choose the architectural context of volunteer computing. For this, we used the XtremWeb-CH environment. Experiments were conducted on a real volunteer computing testbed and show good speed-ups and very acceptable platform overhead, letting XtremWeb-CH be a good candidate for deploying parallel applications over a volunteer computing environment.

## 1 Introduction

The use of distributed architectures for solving large scientific problems seems to become mandatory in a lot of cases. For example, in the domain of radiotherapy dose computation the problem is crucial. The main goal of external beam radiotherapy is the treatment of tumors while minimizing exposure to healthy tissue. Dosimetric planning has to be carried out in order to optimize the dose distribution within the patient. Thus, to determine the most accurate dose distribution during treatment planning, a compromise must be found between the precision and the speed of calculation. Current techniques, using analytic methods, models and databases, are rapid but lack precision. Enhanced precision can be achieved by using calculation codes based, for example, on the Monte Carlo methods. The main drawback of these methods is their computation times which can rapidly become huge. In [18] the authors proposed a new approach, called Neurad, using neural networks. This approach is based on the collaboration of computation codes and multi-layer neural networks used as universal approximators. It provides a fast and accurate evaluation of radiation doses in any given environment for given irradiation parameters. As the learning step is often very time consuming, in [5] the authors proposed a parallel algorithm that enables to decompose

the learning domain into subdomains. The decomposition has the advantage of significantly reducing the complexity of the target functions to approximate.

Now, as there exist several classes of distributed/parallel architectures (supercomputers, clusters, global computing. . .) we have to choose the best suited one for the parallel Neurad application. The volunteer (or global) computing model seems to be an interesting approach. Here, the computing power is obtained by aggregating unused (or volunteer) public resources connected to the Internet. In our case, we can imagine, for example, that a part of the architecture will be composed of some of the different computers of the hospital. This approach presents the advantage of being clearly cheaper than a more dedicated approach like the use of supercomputers or clusters. Furthermore and as we will see in the remainder, the studied parallel algorithm corresponds very well to this computation model.

The aim of this paper is to propose and evaluate a gridification of the Neurad application (more precisely, of the most time consuming part, the learning step) using a volunteer computing approach. For this, we focus on the XtremWeb-CH environment [2]. We chose this environment because it tackles the centralized aspect of other global computing environments such as XtremWeb [11] or Seti [1]. It tends to a peer-to-peer approach by distributing some components of the architecture. For instance, the computing nodes are allowed to directly communicate. Experiments were conducted on a real global computing testbed. The results are very encouraging. They exhibit an interesting speed-up and show that the overhead induced by the use of XtremWeb-CH is very acceptable.

The paper is organized as follows. In Section 2 we present the Neurad application and particularly its most time consuming part, i.e. the learning step. Section 3 details the XtremWeb-CH environment and Section 4 exposes the gridification of the Neurad application. Experimental results are presented in Section 5 and we end in Section 6 by some concluding remarks and perspectives.

## 2 The Neurad Application

The *Neurad* [4] project presented in this paper takes place in a multi-disciplinary project, involving medical physicists and computer scientists whose goal is to enhance the treatment planning of cancerous tumors by external radiotherapy. In our previous works [14,16,18], we have proposed an original approach to solving scientific problems whose accurate modeling and/or analytical description are difficult. That method is based on the collaboration of computational codes and neural networks used as universal interpolator. Thanks to that method, the *Neurad* software provides a fast and accurate evaluation of radiation doses in any given environment (possibly inhomogeneous) for given irradiation parameters. We have shown in a previous work ([5]) the interest of using a distributed algorithm for the neural network learning. We use a classical RPROP [7] algorithm with a HPU [2] topology to do the training of our neural network.

<sup>1</sup> Resilient backpropagation.

<sup>2</sup> High order processing units.

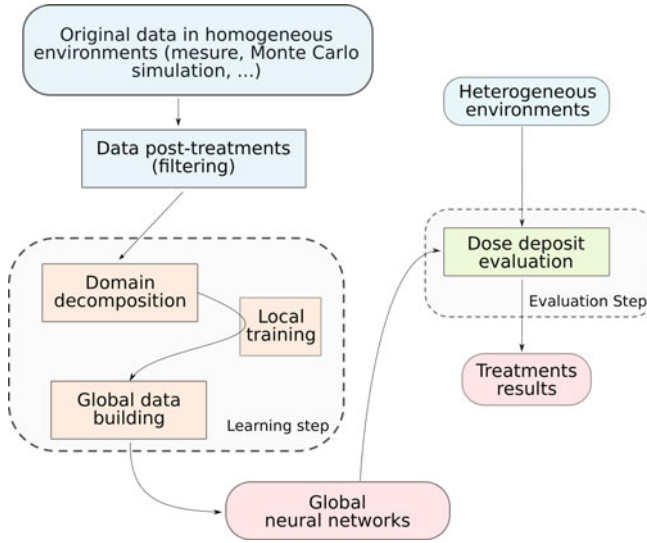
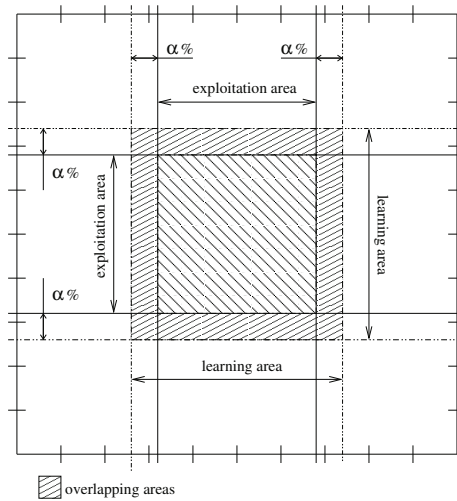


Fig. 1. The Neurad project

Figure 1 presents the *Neurad* scheme. Three parts are clearly independent: the initial data production, the learning process and the dose deposit evaluation. The first step, the data production, is outside of the *Neurad* project. There are many solutions to obtain data about the radiotherapy treatments like the measure or the simulation. The only essential criterion is that the result must be obtained in an homogeneous environment.

The secondary stage of the *Neurad* project is the learning step and this is the most time consuming step. This step is performed offline but it is important to reduce the time used for the learning process to keep a workable tool. Indeed, if the learning time is too huge (for the moment, this time could reach one week for a limited domain), this process should not be launched at any time, but only when a major modification occurs in the environment, like a change of context for instance. However, it is interesting to update the knowledge of the neural network, by using the learning process, when the domain evolves (evolution in material used for the prosthesis or evolution on the beam (size, shape or energy)). The learning time is related to the volume of data which could be very important in a real medical context. Some work has been done to reduce this learning time with the parallelization of the learning process by using a partitioning method of the global dataset. The goal of this method is to train many neural networks on sub-domains of the global dataset. After this training, the use of these neural networks all together allows to obtain a response for the global domain of study.

However, performing the learning on sub-domains constituting a partition of the initial domain may not be satisfying depending on the chosen quality of the results. This comes from the fact that the accuracy of the approximation performed by a neural network is not constant over the learned domain. Thus,



**Fig. 2.** Overlapping for a sub-network in a two-dimensional domain with ratio  $\alpha$

it is necessary to use an overlapping of the sub-domains. The overall principle is depicted in Figure 2. In this way, each sub-network has an exploitation domain smaller than its training domain and the differences observed at the borders are no longer relevant. Nonetheless, in order to preserve the performance of the parallel algorithm, it is important to carefully set the overlapping ratio  $\alpha$ . It must both be large enough to avoid the border's errors, and as small as possible to limit the size increase of the data subsets [5].

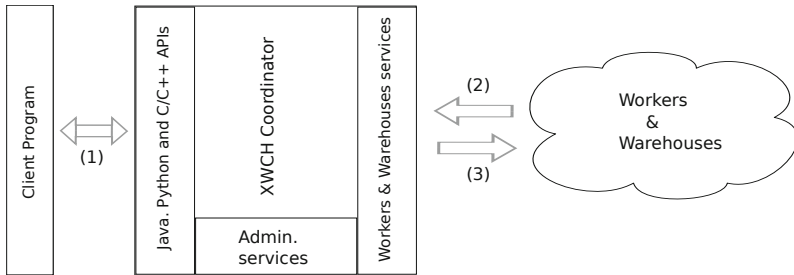
### 3 The XtremWeb-CH Environment

High performance computing environments like MPI (Message Passing Interface) [12] are widely used and have proved their efficiency. This class of systems are very tightly coupled and powerful but not very error tolerant. Cluster computing environments like Condor [17] and volunteer computing systems like BOINC [3] are loosely coupled and have a scheduler that distributes tasks to computing nodes. Cluster computing environments assume the fact that nodes are in general directly accessible, one to another, but this does not apply to volunteer computing systems.

XtremWeb-CH (XWCH) is a volunteer computing inspired large-scale computing platform for distributed applications. In fact, it tends to be a good compromise between cluster computing and volunteer computing. It is originally based on another platform called XtremWeb [11]. It is easy to install, maintain, and it is supported by a Grid middleware (ARC [10]) and a workflow engine (JOpera [9]). It consists in three components: one coordinator, a set of workers, and at least one warehouse. Client programs use these components.

The coordinator is the main component of the XWCH platform. It controls user access and schedules jobs to workers. It provides a web interface for managing jobs and users, and a set of web services. These are user services and worker/warehouse services implemented using WSDL (Web Service Description Language) [8], that simplifies client development for languages that support it (and most popular programming languages do).

A worker is a Java daemon that runs on the user machine. Assumed to be volatile, the workers periodically report themselves to the coordinator, accept jobs, retrieve input, compute jobs, and store the results of the computation on warehouses. If the coordinator does not receive a signal from a worker, it will simply remove it from the scheduling list, and if a job had been assigned to that worker, it will be re-assigned to another one. A schema of the architecture is shown in Figure 3.



**Fig. 3.** The XtremWeb-CH architecture

A warehouse is a file server that acts as a data storage system for workers and client programs. Workers may not necessarily be able to communicate directly with each others, due to firewalls and NAT sub-networks. For these reasons, warehouses are used as intermediaries to exchange, store and retrieve data.

Job submission is done by a client program which is written using a flexible API, available for Java and C/C++ programs. The client program runs on a “client node” and calls the user services to submit jobs (Figure 3, (1)). The main flexibility provided by the use of this architecture is to control and dynamically generate jobs especially when their number cannot be known in advance. Communications between the coordinator and the workers are always initiated by the workers following a pull model (Figure 3, (2)):

- Workers receive jobs (Figure 3, (3)) only if they send a “work request” signal;
- When a worker finishes its job, it stores its output file on a warehouse and sends a “work result” signal to the coordinator;
- During its execution, a worker (respectively warehouse) periodically sends “work alive” to the worker service (respectively warehouse service) to report itself to the coordinator.

As a whole, XWCH is easy to install, maintain and use. Its components are programmed mainly using Java, and their process memory sizes in a typical 32-bit GNU/Linux computer are:

- Coordinator 190 MB including the Glassfish Java container;
- Worker 40 MB;
- Warehouse 80 MB.

Experiments presented in [15] show that the performance of XWCH is comparable with Condor [13], another non-intrusive computing system that has similar functionalities but is somewhat more difficult to install.

The main characteristics of the new version of XWCH, compared to previous ones, are: dynamic job generation, flexible data sharing (data replication) and persistent jobs. These features are presented in [7] and will not be detailed in this paper.

## 4 The Neurad Gridification

As previously exposed, the Neurad application can be divided into three steps. The goal of the first step is to decompose the data representing the dose distribution on an area. This area contains various parameters, like the nature of the medium and its density. This part is out of the scope of this paper.

The second step of the application, and the most time consuming, is the learning in itself. This is the one which has been parallelized, using the XWCH environment. As exposed in section 2, the parallelization relies on a partitioning of the global dataset. Following this partitioning all learning tasks are independently executed in parallel with their own local data part, with no communication, following the fork/join model. Clearly, this computation fits well with the model of the chosen middleware.

The execution scheme is then the following (see Figure 4):

1. We first send the learning application and its data to the middleware. In a first time, we send the application to data warehouses (DW), and we create an "application module" on the coordinator (Coord.) including references retrieved from the previous sending operation. In a second time, we apply the same process to application data.
2. When a worker (W) is ready to compute, it requests a task to execute to the coordinator (Coord.);
3. The coordinator assigns the worker a task. This last one retrieves the application and its assigned data, by requesting them to DW with references sent by the coordinator, and so can start the computation;
4. At the end of the learning process, the worker sends the result to a warehouse.

The last step of the application is to retrieve these results (some weighted neural networks) and exploit them through a dose distribution process. This last step is out of the scope of this paper.

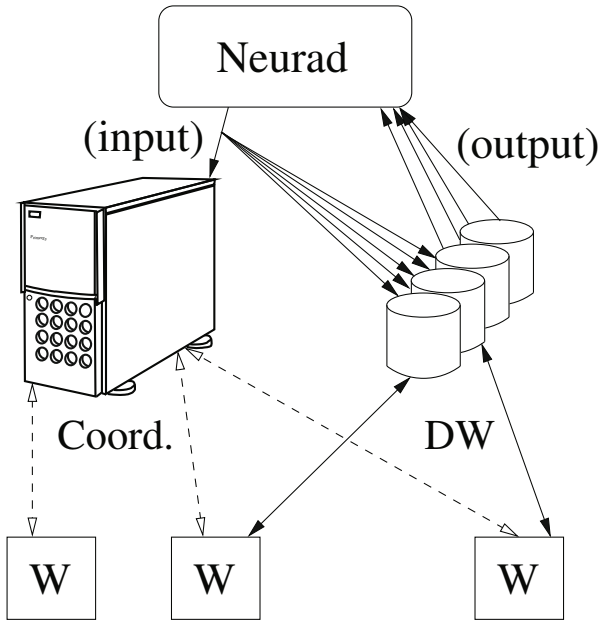


Fig. 4. The proposed Neurad gridification

## 5 Experimental Results

The aim of this section is to describe and analyze the experimental results we have obtained with the parallel Neurad version previously described. Our goal was to carry out this application with real input data and on a real volunteer computing testbed.

**Experimental conditions.** The size of the input data is about 2.4Gb. In order to avoid noises to appear and disturb the learning process, these data can be divided into, at most, 25 parts. This generates input data parts of about 15Mb (in a compressed format). The output data, which are retrieved after the process, are about 30Kb for each part. We used two distinct deployments of XWCH:

1. In the first one, called “distributed XWCH”, the XWCH coordinator and the warehouses were located in Geneva, Switzerland while the workers were running in the same local cluster in Belfort, France.
2. The second deployment, called “local XWCH” is a local deployment where coordinator, warehouses and workers were, in the same local cluster, at the same time.

For both deployments, the local cluster is a campus cluster and during the day these machines were used by students of the Computer Science Department of the IUT of Belfort. Unfortunately, the data decomposition limitation does not allow us to use more than 25 computers (XWCH workers).



In order to evaluate the overhead induced by the use of the platform we have furthermore compared the execution of the Neurad application with and without the XWCH platform. For the latter case, we want to insist on the fact that the testbed consists only in workers deployed with their respective data by the use of shell scripts. No specific middleware was used and the workers were in the same local cluster.

Finally, five computation precisions were used:  $1e^{-1}$ ,  $0.75e^{-1}$ ,  $0.50e^{-1}$ ,  $0.25e^{-1}$ , and  $1e^{-2}$ .

**Results.** Table 1 presents the execution times of the Neurad application on 25 machines with XWCH (local and distributed deployment) and without XWCH. These results correspond to the measures of the same steps for both kinds of execution, i.e. the sending of local data and the executable, the learning process, and retrieving the results. Results represent the average time of 5 executions.

**Table 1.** Execution time in seconds of the Neurad application, with and without using the XWCH platform

Precision	1 machine	Without XWCH	With XWCH	With local XWCH
$1e^{-1}$	5190	558	759	629
$0.75e^{-1}$	6307	792	1298	801
$0.50e^{-1}$	7487	792	1010	844
$0.25e^{-1}$	7787	791	1000	852
$1e^{-2}$	11030	1035	1447	1108

As we can see, in the case of a local deployment the overhead induced by the use of the XWCH platform is about 7%. It is clearly a low overhead. Now, for the distributed deployment, the overhead is about 34%. Regarding the benefits of the platform, it is a very acceptable overhead which can be explained by the following points.

First, we point out that the conditions of executions are not really identical between, with and without, XWCH contexts. For this last one, though the same steps were achieved out, all transfer processes are inside a local cluster with a high bandwidth and a low latency. Whereas when using XWCH, all transfer processes (between datawarehouses, workers, and the coordinator) used a wide network area with a smaller bandwidth. In addition, in the executions without XWCH, all the machines started immediately the computation, whereas when using the XWCH platform, a latency is introduced by the fact that a computation starts on a machine, only when this one requests a task.

This underlines that, unsurprisingly, deploying a local coordinator and one or more warehouses near a cluster of workers can enhance computations and platform performances.

## 6 Conclusion and Future Works

In this paper, we have presented a gridification of a real medical application, the Neurad application. This radiotherapy application tries to optimize the irradiated dose distribution within a patient. Based on a multi-layer neural network, this application presents a very time consuming step, i.e. the learning step. Due to the computing characteristics of this step, we have chosen to parallelize it using the XtremWeb-CH volunteer computing environment. Obtained experimental results show good speed-ups and underline that overheads induced by XWCH are very acceptable, letting it be a good candidate for deploying parallel applications over a volunteer computing environment.

Our future works include the testing of the application on a larger scale testbed. This implies, the choice of a data input set allowing a finer decomposition. Unfortunately, this choice of input data is not trivial and relies on a large number of parameters.

We are also planning to test XWCH with parallel applications where communication between workers occurs during the execution. In this way, the use of the asynchronous iteration model [6] may be an interesting perspective.

## Acknowledgements

This work was supported by the European Interreg IV From-P2P project.

## References

1. Seti@home, <http://setiathome.ssl.berkeley.edu>
2. Abdennadher, N., Boesch, R.: Towards a peer-to-peer platform for high performance computing. In: Cérin, C., Li, K.-C. (eds.) GPC 2007. LNCS, vol. 4459, pp. 412–423. Springer, Heidelberg (2007)
3. Anderson, D.P., Fedak, G.: The computational and storage potential of volunteer computing. In: IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE Press, Los Alamitos (May 2006)
4. Bahi, J.M., Contassot-Vivier, S., Makovicka, L., Martin, E., Sauget, M.: Neurad. Agence pour la Protection des Programmes, no: IDDN.FR.001.130035.000.S.P.2006.000.10000 (2006)
5. Bahi, J.M., Contassot-Vivier, S., Sauget, M.: An incremental learning algorithm for functional approximation. *Advances in Engineering Software* 40(8), 725–730 (2009), doi:10.1016/j.advengsoft.2008.12.018
6. Bahi, J.M., Couturier, R., Laiymani, D.: Comparison of conjugate gradient and multisplitting algorithms of nas benchmark with the jace environment. In: IPDPS 2008. IEEE Computer Society Press, Los Alamitos (April 2008)
7. Ben Belgacem, M., Abdennadher, N., Niinimaki, M.: Virtual ez grid: A volunteer computing infrastructure for scientific medical applications. In: Bellavista, P., Chang, R.-S., Chao, H.-C., Lin, S.-F., Sloat, P.M.A. (eds.) GPC 2010. LNCS, vol. 6104, pp. 385–394. Springer, Heidelberg (2010)
8. Cerami, E.: *Web Services Essentials – Distributed Applications with XML-RPC, SOAP, UDDI and WSDL*. O’Reilly, Sebastopol (2002)

9. Alonso, G., Pautasso, C.: Jopera: a toolkit for efficient visual composition of web services. *International Journal of Electronic Commerce (IJEC)* (2005)
10. Ellert, M., Gronager, M., Konstantinov, A., Konya, B., Lindemann, J., Livenson, I., Nielsen, J.L., Niinimäki, M., Smirnova, O., Waananen, A.: Advanced resource connector middleware for lightweight computational grids. *Future Generation Computer Systems* 23(2), 219–240 (2007)
11. Fedak, G., Germain, C., Neri, V., Cappello, F.: Xtremweb: A generic global computing system. In: *IEEE International Symposium on Cluster Computing and the Grid*, vol. 0, p. 582 (2001)
12. Gropp, W., Lusk, E., Skjellum, A.: *Using MPI: portable parallel programming with the message passing interface*. MIT Press, Cambridge (1994)
13. Litzkow, M.J., Livny, M., Mutka, M.W.: Condor-a hunter of idle workstations. In: *Condor-a Hunter of Idle Workstations*. IEEE, Los Alamitos (1988)
14. Makovicka, L., Vasseur, A., Sauget, M., Martin, E., Gschwind, R., Henriët, J., Salomon, M.: Avenir des nouveaux concepts des calculs dosimétriques basés sur les méthodes de Monte Carlo. *Radioprotection* 44(1), 77–88 (2009)
15. Niinimäki, M., Ben Belcagem, M., Abdennadher, N.: Xwch-ccgrid. Technical report (2011)
16. Sauget, M., Laurent, R., Henriët, J., Salomon, M., Gschwind, R., Contassot-Vivier, S., Makovicka, L., Soussen, C.: Efficient domain decomposition for a neural network learning algorithm, used for the dose evaluation in external radiotherapy. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) *ICANN 2010*. LNCS, vol. 6352, pp. 261–266. Springer, Heidelberg (2010)
17. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience* 17(2-4), 323–356 (2005)
18. Vasseur, A., Makovicka, L., Martin, E., Sauget, M., Contassot-Vivier, S., Bahi, J.M.: Dose calculations using artificial neural networks: a feasibility study for photon beams. *Nucl. Instr. and Meth. in Phys. Res. B* 266(7), 1085–1093 (2008)

# Leasing Service for Networks of Interactive Public Displays in Urban Spaces

Marko Jurmu<sup>1</sup>, Hannu Kukka<sup>1</sup>, Simo Hosio<sup>1</sup>, Jukka Riekk<sup>2</sup>, and Sasu Tarkoma<sup>3</sup>

<sup>1</sup>MediaTeam Oulu

<sup>2</sup>Intelligent Systems Group, Department of Electrical and Information Engineering, University of Oulu, Finland  
{firstname.lastname}@ee.oulu.fi

<sup>3</sup>Helsinki Institute for Information Technology HIIT,  
Helsinki University of Technology, Espoo, Finland  
sasu.tarkoma@hiit.fi

**Abstract.** This paper reports performance metrics and end-user perceptions of a display leasing service, introduced for a network of public displays. Through leasing, users can acquire a temporary ownership of a public display and utilize it as an application UI element together with a personal mobile device. We argue that this functionality is necessary as the quantity of public displays grows, in order to facilitate public resource utilization and alleviate patterns such as queuing and polling. We implemented a prototype of the leasing service and conducted a field trial with a real-world city centre scenario, through which we analyze the service's signaling overhead and key deployment challenges.

**Keywords:** urban computing, device ensemble, resource discovery, application composition, negotiation.

## 1 Introduction and Related Work

Urban areas are witnessing a large-scale emergence of *interactive public displays*. These displays come in various form factors and sizes, may be installed vertically or horizontally and offer high visual capacity for digital signage as well as interactive content. In other words, there exists a large potential for physically distributed and composable applications within this emergence [1, 2].

Realizing this potential, however, entails few core technical challenges. First, the displays need support for *functional coupling* before users can utilize the display as an additional UI element in their application structures. This fundamental requirement has led to research on *device pairing* [3, 4], where a public display is logically coupled with user's personal mobile device. During the pairing, users can influence the display's behavior through both control and content from personal mobile devices [5].

Device pairing, however, is only intended for joining two devices together as a logical whole. We argue that display utilization based solely on pairing can become a clear bottleneck when the amount of displays grows, as besides current social conventions of queuing and polling, users have no way of inferring the status information of a certain target display. In addition, there might be available displays

close-by, but if they are not in line-of-sight, users cannot discover them. We refer to this setting as *one display, many users*.

For these reasons, the utilization of an interactive public display within distributed applications needs to be managed with *an explicit notion of temporary ownership*, which results from a *negotiation* phase between a personal mobile device and a public display. In addition, the displays need to be *interconnected* in order to enable two fundamental services: First is the ability for a mobile device to *query* the network of displays as *one logical whole* and thus gain an integrated view of display utilization. The dynamic status information augments static display properties such as location, resolution etc. Second, the displays need to communicate with each other in cases where ownerships need to be scheduled from one display to another, such as application session transfer. Naturally, local scheduling and queuing of ownerships is required as well. In this setting, mobile devices will first query the network, select a target display, negotiate with the target and finally conduct a pairing on-site. We refer to this setting as *many displays, many users*.

Related work in this area have focused mainly on device pairing solutions and subsequent organization of application presentation and input mappings [6, 7, 8, 9]. Main difference to our work is that we solely focus on the modeling and management of display ownership and status information within an interconnected network of public displays. In our previous work, we have reported one solution for visualizing the resource discovery information in mobile devices [10] in case of internetworked displays. We have also suggested the need for the discovery process to be augmented with a negotiation phase prior to application composition and usage [11]. Finally, we have reported how the application data tier can be separated from the runtime presentation tier in case of applications utilizing both public displays and mobile devices [12].

In this paper we present an implementation and evaluation findings of a display leasing service, aimed as a solution to the negotiation phase of utilizing a public display. We set out to investigate the feasibility of the leasing service by implementing a functional prototype and testing it in a laboratory setting. Through this testing, we wanted to ensure that the real-time performance of the leasing service was sufficient for a subsequent field trial. Laboratory development was done in spring 2009, and the prototype was deployed as part of a field study in June 2009. This study ran until end of August 2009, and in this paper we focus on data collected during a period of 46 days from July 17 to August 31.

Contributions of this work are as follows:

- Presenting the concept of display leasing as a solution to the negotiation functionality required by physically distributed and composable applications.
- Evaluating a prototype of the leasing service in laboratory environment.
- Analyzing findings from a deployment of the leasing service as part of a field trial of interactive public displays in downtown area of City of Oulu, Finland.

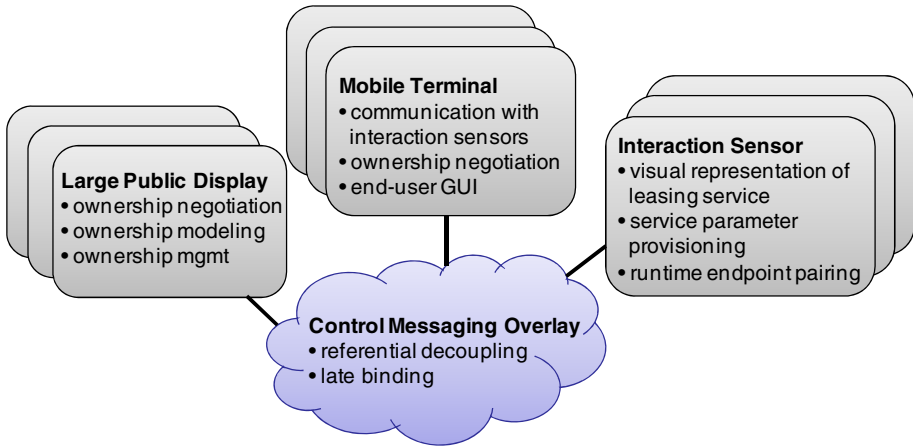
## 2 Prototype System

As outlined in the introduction, we consider the leasing service within a network of public displays. Figure 1 illustrates this network with the associated entities. On a

high-level, the architecture consists of *large (interactive) public displays*, *mobile terminals*, *interaction sensors* and a *control messaging overlay*. Within the focus of this paper, each display has independent middleware-level software component, which manages the display's temporary ownerships through negotiation, modeling and management processes.

Mobile terminals feature middleware which communicates with the interaction sensors, negotiates with displays through the messaging overlay and visualizes negotiation status information to user. Interaction sensors are spatially associated with the displays. Their purpose is to visually advertise the leasing possibility for users, resolve the target display to the mobile terminal through service parameters and initiate the negotiation phase with a tangible action of touch. We have not incorporated remote discovery functionality in this architecture, as we wanted to support social, 'arena' type of applications where several users can join easily on-site through interaction sensors. Exclusiveness of the leased ownership can be specified in the metadata of the lease during the negotiation.

The control messaging overlay is designed to support referential decoupling in order to allow topic-based query messages within the network such as discovery, which is out of the scope of this paper. In addition, the late binding feature allows endpoints to establish one-to-one connections only when necessary, thus removing the need for sequential polling of target displays.



**Fig. 1.** Conceptual architecture. The messaging overlay joins large public displays, mobile terminals and interaction sensors to enable runtime discovery, negotiation and pairing.

Implementation of the conceptual architecture is depicted in Figure 2. Public displays are each equipped with touch screens and dedicated control PCs that host the respective middleware components. *Resource Management* implements the leasing and discovery services of the display, including modeling and management of leases. *Layout Manager* manages the screen real estate of the display. *RFID Reader Management* implements the interaction sensor functionality. *Face Detection* software is out of this paper's scope.

Mobile terminals implement the client sides of discovery and leasing services, and feature dedicated launcher software for starting distributed applications [12]. In addition, each mobile terminal has an associated RFID tag with its unique ID linked to the terminal by a lookup table in the middleware backend server. Similar lookup table associates each interaction sensor to one dedicated display. See Figure 3 for illustrations of RFID accessories utilized in the field trial.

Control messaging overlay is implemented through open source Fuego [13] publish/subscribe system with content-based routing. Displays subscribe topics related to *leasing* and *discovery* from the messaging, and include their pub/sub endpoint addresses in the query responses. These endpoint addresses have been implemented with a flat ID address scheme; they are thus separate from the actual IP addresses of the displays.

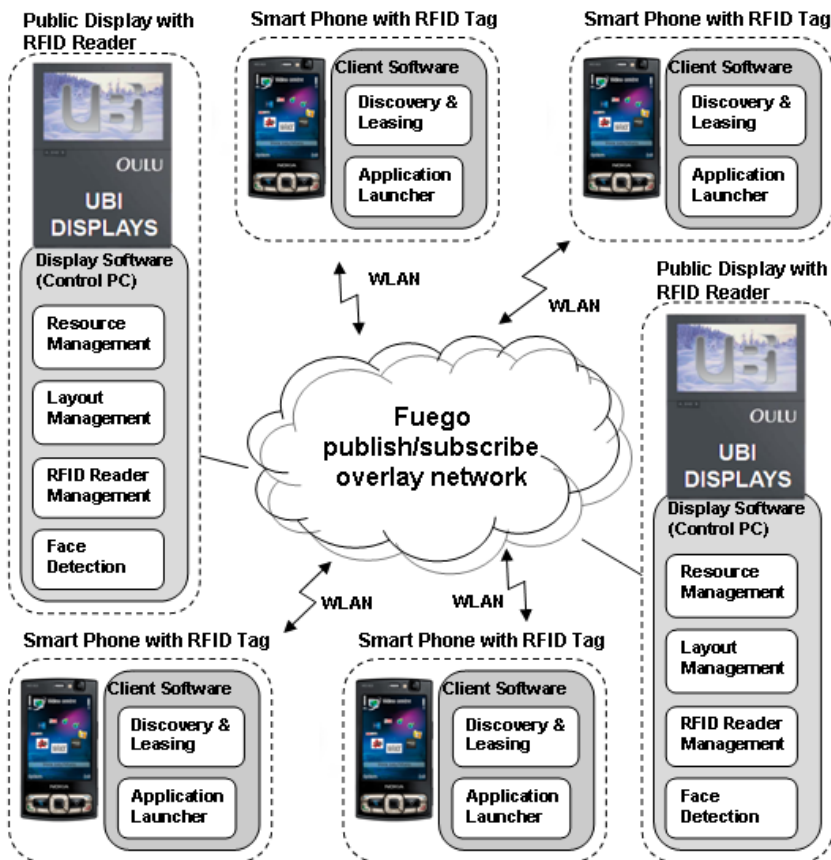


Fig. 2. Prototype implementation

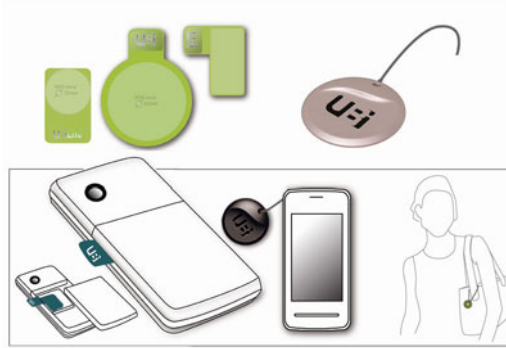


Fig. 3. Two designs of mobile terminal RFID accessory

## 2.1 Leasing Sequence

Leasing sequence starts when user touches the interaction sensor of a target display with his/her mobile terminal's RFID accessory, which leads to a runtime pairing of the mobile terminal and the target display. The sequence is a series of point-to-point plaintext RPC messages communicated on top of the publish/subscribe overlay, as the endpoints are resolved either during query messages, or RFID touch event processing. The sequence contains the following control primitives:

- `lease_request`, sent by the mobile terminal and replied by the display with `lease_response`.
- `launch_request`, sent by the mobile terminal after a granted lease and selection of application, replied by the display with `launch_response`.
- `termination_request`, sent by the mobile terminal when the application is closed, replied by the display with `termination_response`.

Each of the phases in this sequence is initiated by the user, and after each phase the user is presented with options regarding continuation, through the mobile terminal GUI [12]. After the lease is granted, user is notified on the mobile terminal screen to choose an application to start, which then leads to launching phase. Subsequently, closing the application instructs the mobile terminal to initiate the termination phase.

The lease request phase of the sequence consists of the following steps: *authenticating* the request against the local access control list of the display, *construction* of a new lease with information of the requesting mobile terminal, and *scheduling* of this lease within the management of the display.

The launch request phase parameterizes the application ID chosen by the user to the display, and both endpoints subsequently proceed to launch the respective parts of the distributed application's presentation. The launch termination phase is initiated when the mobile side application presentation closes, and it instructs the display to unload the application presentation(s) rendered on the display.



## 2.2 Lease Types and Local Scheduling

The scheduling step places the lease into a FIFO queue on the display, and in case the display already features an *active lease*, i.e., a lease currently utilized by other user(s), new lease is placed into the queue as a *pending lease*. This queuing status is then prompted to the user in the mobile terminal GUI after receiving the lease response. When the active lease ends, the display management software schedules the next pending lease from the FIFO queue as the active lease. Scheduling causes a message to be sent to the corresponding mobile terminal which then actuates a haptic notification (vibration alert) to the user, indicating the promotion of the lease as active. This message is automatically subscribed if the lease is scheduled as pending.

In case the active lease is typed as *social lease*, the incoming request is interpreted as a join attempt to the active lease. In this case, the only application choice prompted to the user in the launch phase is the application currently running within the active social lease. Social application structure is closed when the last owner of the social lease terminates his/her application presentation. As a conclusion, social leases cannot be queued in this implementation. This was a clear design point we wanted to make in this prototype, indicating the emphasis of social applications with multiple mobile terminals simultaneously participating to the application structure on-site.

## 3 Evaluation

### 3.1 Laboratory Experiments

Testing of the leasing service prototype was conducted in a laboratory environment by utilizing one public display as a target, and subjecting it to thirty (30) independent leasing negotiations including all the phases outlined in section 2.1. The display control PC features a dual-core CPU, 4GB of RAM and 100 Mb/s IEEE 802.3 Ethernet connectivity. Mobile terminal utilized in the lab tests is a Nokia model N95 8GB, which communicated with the display through IEEE802.11g WLAN-network.

From these measurements, mobile terminal side measured the overall latency of each phase through timers implemented inline to the middleware code. Timer started prior to publishing the lease request message, and stopped after the reply message had been received and processed. We implemented dedicated timers for each RPC call, as there are user decisions made through the mobile GUI within the sequence, and we didn't want them to bias the latencies.

### 3.2 Field Testing

Field testing in downtown Oulu was organized within a network of public display which is one part of a larger field trial installation. A field trial office was set up to downtown, where potential users could register to the system. Through this registration, users could loan a mobile terminal with an RFID accessory and the associated middleware software pre-installed. If a user wanted to utilize his/her own smart phone with sufficient capabilities, the field trial office took care of installing the required software to the phone and attaching the RFID accessory of user's choice.

During the field trial, the installation of public displays consisted of 6 indoor displays in locations such as main library, lobby area of a large municipal swimming hall, a youth culture centre, as well as 6 double-sided outdoor displays covering the marketplace area as well as the main pedestrian street of Oulu, called Rotuaari. In addition, the Oulu downtown area is blanketed with an open municipal IEEE 802.11g WLAN network, which the mobile terminals utilized during the field trial. Finally, each of the displays was also fitted with an IEEE 802.11g access point, ensuring the WLAN coverage around the displays.

During the field trial, we had personnel conducting *tutorial sessions* for the purpose of *educating end-users* and *managing end-user expectations* regarding the installation. These tutorials were conducted at certain time of each week, next to the outdoor display in the marketplace area. Through these tutorial sessions, we hoped to clarify the idea of the distributed applications and the display leasing for the users, and to popularize the display installation as a whole. In addition, the field trial office personnel conducted a series of *semi-structured interviews*. An overview of the public display installation can be found in [14].

## 4 Results

### 4.1 Lab Experiments

As outlined in section 2.1, the setup phase of the leasing sequence consists of lease and launch phases. When these phases are complemented with endpoint resolving of the mobile terminal based on an RFID touch event, the complete setup latency consists of the following components:

$$T_{total} = T_{rfid} + T_{lease\_request} + T_{launch\_request} \quad (1)$$

Here,  $T_{total}$  is the complete setup latency,  $T_{rfid}$  is the interaction sensor-based lookup operation,  $T_{lease\_request}$  is the lease requesting phase and  $T_{launch\_request}$  is the application setup phase. Latency  $T_{total}$  thus starts from the touch of the interaction sensor, and ends when the application structure is launched. It should be noted that user is prompted for application selection prior to launch request, and since this is a subjective selection task, it was not included in the latency components.

Table 1 lists the individual latency components of a leasing sequence implemented in the prototype. These values were acquired by conducting thirty (30) independent leasing sequences, and calculating the average and standard deviation from each phase. It should be noted that the mobile terminal's endpoint lookup in time  $T_{rfid}$  is implemented with a pure TCP socket connection, thus making it significantly faster than the subsequent messages.

**Table 1.** Latency components of the setup part of the leasing sequence

Latency Component	Average (ms)	Standard Deviation (ms)
Target resolving	26	7.2
Lease request	473	103
Launch request	260	47

From table 1, it becomes clear that the lease request phase is the largest individual component in the overall latency. This is due to processing done in the display side, as explained in section 2.1. From the average latencies, we can infer that the overall signaling overhead for setting up a leased application session is in the order of 700 ms – 800 ms. We did not conduct measurements on the lease termination phase, as this latency is directly dependent on the active lease type and required scheduling on the display side. We estimate, however, that for each individual mobile terminal the termination latency in the worst case is in the order of the lease request delay.

Table 2 shows the payloads of individual messages exchanged during the leasing sequence. The message sizes are in accordance with the leasing sequence design, where messages consist of control primitive, addressing information and possibly additional payload such as lease typing and queue status. Publish/subscribe overlay utilizes a three-tuple model (`<key, value, value_type>`) in encoding key-value pairs for messages.

**Table 2.** Payloads of individual messages exchanged in the leasing session setup

Message Type	Message Size (B)
Lease request	433
Lease response	765
Launch request	592
Launch response	661
Termination request	533
Termination response	376

## 4.2 Field Testing

During the field testing, the leasing service was tested in total by 80 users, either with a mobile device of their own or one loaned from the field trial office. Altogether, the displays were leased 726 times during the data collection period. In case of social leases, each lease is considered individually in the overall amount. The durations of the leased interaction sessions had an average of 1:33 minutes, which is in accordance with the types of applications [12] that were developed and utilized on top of this service. Reflecting the latency measurement to this duration yields a signaling overhead percentage of  $(26\text{ms} + 473\text{ms} + 260\text{ms} + \sim 500\text{ms}) / 93000\text{ms} * 100\% = 1.35\%$ . Detailed description of the individual applications is out of this paper's scope.

Within the semi-structured interviews, users were asked – among other questions – if they would like to use the displays in conjunction with their personal mobile devices and why/why not. This interview data served as recording the first impressions of citizens towards the displays. Some example answers are listed below:

- *It would be nice to download maps or bus schedules from the display to mobile device for later checking. Female, 17.*
- *I'd use them (services to mobile) if they are made easy enough. If too many steps are involved, I'm not going to bother. Male, 19.*
- *I doubt that I would use this [functionality]; I'm not accustomed to use my mobile device that way. Female, 25.*

- *I mostly use my mobile device for calls and SMS. But if this functionality would generalize and be simple to use, I might consider it. Female, 26.*
- *As long as these services would be free, I would use them. Male 26.*

## 5 Discussion and Future Work

This paper motivated, outlined and evaluated a service for end users to gain temporary ownerships of public displays through their personal mobile devices. As this type of service – especially in urban spaces – is novel to our knowledge, there are multiple open issues and challenges related to it. We however see it as a necessary addition to the already researched device pairing functionality in this context, as it will significantly facilitate the discovery and allocation of UI resources required by physically distributed and composable applications.

In addition to the latency measurements, majority of our insights of the prototype came from the field test. First, if the latency measurements acquired in the lab testing are compared to the average interaction times of the distributed applications during the field test, we can conclude that the leasing overhead is in the order of two percent, which we consider a reasonable delay. We consider this overhead acceptable, but also acknowledge that this is the first version of the leasing prototype with several straightforward assumptions such as simplified authentication and a relatively simple queue management. When considering the current payload sizes of the messages, this negotiation is a light load for the publish/subscribe architecture, and no major scalability issues should exist. During the field trial, communication latencies of lease negotiations varied from those measured in the lab to several seconds, and in some cases the mobile terminals dropped the WLAN connection without prior notice, making the evaluation challenging.

Perhaps the single most significant finding from the field trial was the persistence of existing mental models of the end-users towards this type of solution for public resource usage. Despite the regular tutorial sessions arranged on-site during the field trials, users were having a hard time conceiving the principle of this prototype. A good indicator of this is that the queuing functionality built to the prototype in the lab was actually never used in the field trial. In addition, the interview data indicated that while users saw benefits in combining personal mobile devices with the displays, they almost exclusively referred to so-called *information pick-up* services, where certain information is transferred from the display to the mobile device for private viewing at a later time.

We mostly account this to social conventions concerning the displays as public resources, whereas users either conceived the displays as pure advertising screens, or did not want to disturb others currently occupying the resource. From this viewpoint, the initiation of the leasing sequence purely through the on-site interaction sensors was a design weakness. The difference of the implemented solution to more common ways of public queuing in places such as banks etc. is that the queues are by design distributed across the display network, more equivalent to supermarket style of queue distribution. This brings challenges to how people perceive and assess individual queues of the displays, and these need to be incorporated in future design iterations.

Our opinion is that further investigations of this service need to be conducted in more rigidly controlled user tests, rather than subjecting them to semi-controlled field studies. This is because the end-user education and factors related to existing mental models need to be better contained within the study. We're currently planning a similar but smaller installation of displays to campus environment for these reasons, while other services of the field trial continue to evolve in the downtown displays.

There are multiple features in this negotiation solution that require further investigation: First, the granting of the lease is covered with one RPC call, which makes assumptions regarding the capabilities of the target display. In more general case, this phase needs to be extended with messaging where mobile terminal receives the display capabilities and replies with certain *suitability metric* back to the display. In our case, the capabilities of each display are identical and known in design time, so this part of the negotiation was not required. In addition, *systematic modeling* of public display capabilities in conjunction with the dynamic usage information is required in order to further pursue this solution.

One interesting problem arising from this testing is the scheduling of leasing queues containing a mixture of exclusive and social leases. Related to this, it is yet unknown how much control of the scheduling process should be given to the end-users contending from the display, as we suspect that the fluid social conventions around the display [15] heavily influence this contention. Already now it seems clear that this functionality cannot be exclusively the responsibility of either system or the end-users, but a method of dialogue is required.

Finally, the current leasing sequence does not incorporate any security measures, which will be crucial in the long run. Security can be brought to this service by encrypting the publish/subscribe messages, and implementing the access controls either based on existing ACL solutions, or by utilizing a 3<sup>rd</sup> party authentication server.

## References

1. Boring, S., Baur, D., Butz, A., Gustafson, S., Baudisch, P.: Touch Projector: Mobile Interaction Through Video. In: 28th International Conference on Human Factors in Computing Systems (CHI 2010), pp. 2287–2296. ACM, New York (2010)
2. Pering, T., Want, R., Rosario, B., Sud, S., Lyons, K.: Enabling Pervasive Collaboration with Platform Composition. In: Tokuda, H., Beigl, M., Friday, A., Brush, A., Tobe, Y. (eds.) Pervasive 2009. LNCS, vol. 5538, pp. 184–201. Springer, Heidelberg (2009)
3. Pering, T., Ballagas, R., Want, R.: Spontaneous Marriages of Mobile Devices and Interactive Spaces. *Commun. ACM* 48(9), 53–59 (2005)
4. Rukzio, E., Leichtenstern, K., Callaghan, V., Holleis, P., Schmidt, A., Chin, J.: An Experimental Comparison of Physical Mobile Interaction Techniques: Touching, Pointing and Scanning. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 87–104. Springer, Heidelberg (2006)
5. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G.: The Smart Phone: A Ubiquitous Input Device. *IEEE Pervasive Computing Journal* 5(1), 70–77 (2006)

6. Soroker, D., Paik, Y.S., Moon, Y.S., McFaddin, S., Narayanaswami, C., Jang, H.K., Coffman, D., Lee, M.C., Lee, J.K., Park, J.W.: User-Driven Visual Mashups in Interactive Public Spaces. In: 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2008), ICST, Brussels, article #43 (2008)
7. Storz, O., Friday, A., Davies, N., Finney, J., Sas, C., Sheridan, J.G.: Public Ubiquitous Computing Systems: Lessons Learned from the e-Campus Display Deployments. *IEEE Pervasive Computing Journal* 5(3), 40–47 (2006)
8. Sharp, R., Madhavapeddy, A., Want, R., Pering, T.: Enhancing Web Browsing Security on Public Terminals Using Mobile Composition. In: 6th International Conference on Mobile Systems, Applications and Services (MobiSys 2008), pp. 94–105. ACM, New York (2008)
9. Rellermeyer, J.S., Riva, O., Alonso, G.: AlfredO: An Architecture for Flexible Interaction with Electronic Devices. In: Issarny, V., Schantz, R. (eds.) *Middleware 2008*. LNCS, vol. 5346, pp. 22–41. Springer, Heidelberg (2008)
10. Jurmu, M., Boring, S., Riekkii, J.: ScreenSpot: Multidimensional Resource Discovery for Distributed Applications in Smart Spaces. In: 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2008). ICST, Brussels, article #41 (2008)
11. Jurmu, M., Perttunen, M., Riekkii, J.: Lease-Based Resource Management in Smart Spaces. In: 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 622–626. IEEE Press, New York (2007)
12. Hosio, S., Jurmu, M., Kukka, H., Riekkii, J., Ojala, T.: Supporting Distributed Private and Public User Interfaces in Urban Environments. In: 11th Workshop on Mobile Computing Systems and Applications (HotMobile 2010), pp. 25–30. ACM, New York (2010)
13. Tarkoma, S., Kangasharju, J., Lindholm, T., Raatikainen, K.: Fuego: Experiences with Mobile Data Communication and Synchronization. In: *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2006)*, Helsinki, Finland, pp. 1–5 (2006)
14. Ojala, T., Kukka, H., Lindén, T., Heikkinen, T., Jurmu, M., Hosio, S., Kruger, F.: UBI-Hotspot 1.0: Large-Scale Long-Term Deployment of Interactive Public Displays in a City Center. In: 5th International Conference on Internet and Web Applications and Services (ICIW 2010), pp. 285–294. IEEE Computer Society, Los Alamitos (2010)
15. Peltonen, P., Kurvinen, E., Salovaara, A., Jacucci, G., Ilmonen, T., Evans, J., Oulasvirta, A., Saarikko, P.: It’s Mine, Don’t Touch!: Interactions at a Large Multi-Touch Display in a City Centre. In: 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI 2008), pp. 1285–1294. ACM, New York (2008)

# Yarta: A Middleware for Managing Mobile Social Ecosystems\*

Alessandra Toninelli, Animesh Pathak, and Valérie Issarny

INRIA Paris Rocquencourt, France

{alessandra.toninelli,animesh.pathak,valerie.issarny}@inria.fr

**Abstract.** With the increased prevalence of advanced mobile devices (the so-called “smart” phones), interest has grown in mobile social ecosystems, where users not only access traditional Web-based social networks using their mobile devices, but are also able to use the context information provided by these devices to further enrich their interactions. In complex mobile social ecosystems of the future the heterogeneity of software platforms on constituent nodes, combined with their intrinsic distributed nature and heterogeneity of representation of data and context raises the need for middleware support for the development of mobile social applications.

In this paper, we propose Yarta, a novel middleware designed for mobile social ecosystems (MSE), which takes into account the heterogeneity of both deployment nodes and available data, the intrinsic decentralized nature of mobile social applications, as well as users’ privacy concerns. To validate our approach, we show how we developed two mobile social applications over Yarta, and report on both its efficiency and ease-of use by way of extensive evaluation on smart phones and laptops.

## 1 Introduction

Social ties such as friendship, common interests, and shared professional activities are central to humans as these ties bind individuals together. This web of social bindings is referred to as a *social network*, and is the focus of so-called *social applications*, i.e., applications that support human social interactions and are characterized by their swarming, transitory, and informal qualities [4, 8]. Recent advances in wireless network technologies and the increasing diffusion of smart phones equipped with sensing capabilities represent a unique chance to enhance social applications and make them truly pervasive in everyday life [4, 14].

A salient feature of these situations is that physical places can also act as social filters. For example, a conference venue groups together attendees belonging to different organizations, coming together and socially interacting in a number of ways, such as listening to and making comments on talks and presentations, setting up scheduled and spontaneous meetings, exchanging technical knowledge and extending their professional network. Similarly, people attending a football

---

\* Funded in part by an ERCIM *Alain Bensoussan* Fellowship Programme project.

match generally share an interest in football and sport, and support the same team, especially if they find themselves in physical proximity. We propose the term *mobile social ecosystem* (MSE) to describe this richer set of interactions occurring between the participants in these situations. In MSEs, the heterogeneity of software platforms on constituent nodes, combined with their intrinsic distributed nature and heterogeneity of representation of data and context raises the need to support the development of *mobile social applications* (MSAs).

As discussed in the next section, several MSAs and supporting middlewares have emerged recently. However, developing mobile social applications is still a challenging task for several reasons:

**Lack of consistent API for social sensors.** Today, application developers need to directly interface with sources of social information (also known as *social sensors*), such as the user's contacts list or call log on the phone, or his personal or professional profile in online social networking sites. Although popular social applications, such as LinkedIn<sup>1</sup> or Facebook<sup>2</sup>, have recently allowed to export the user's profile in a standard format, a common platform to mediate access to all social data by different applications is still lacking.

**App-specific data silos.** Related to the above, the social data collected by today's MSAs are not designed to be accessible to other MSAs. E.g, Facebook owns users' data that might be reused by other applications only if those applications are integrated into it. Similar considerations hold for most mobile social applications, such as [12] and [1]. As a result, current mobile social applications produce and manage their own data, which can be imported to other applications only with considerable effort, raising issues such as semantic consistency.

**Lack of advanced social access control mechanisms.** Since MSAs manage contextual data such as mobility traces, user preferences and activities, which are sensitive per se and can be further used to infer sensitive information, it raises critical security issues, particularly in terms of privacy and access control of users' data. The task becomes even more difficult given the networked nature of mobile social applications, where information comes from multiple sources, moves to multiple destinations (possibly unforeseen at information production time) and is linked to other information following unpredictable patterns. Recent solutions have shed light on these issues, and made a relevant, albeit only initial, effort to tackle the problem [5].

**Dependence on centralized solutions.** Ubiquitous environments are naturally decentralized, and users must be able to access their social data anywhere and anytime, regardless of access to the Internet. In addition, a global view of users' MSE may not (always) be available, while privacy and portability issues might discourage approaches based on MSE data replication on each user's device [5]. Centralized architectures used by current Web social applications and platforms, thus, are not appropriate for the mobile setting, nor their extensions

---

<sup>1</sup> <http://linkedin.com>

<sup>2</sup> <http://facebook.com>



to support access from mobile devices, which always rely on some server to store data and manage users' interactions.

To address the above challenges, in this paper we propose Yarta, a novel middleware support platform for mobile social applications (or an *MSE management middleware*). In particular, (i) Yarta is based on an expressive and extensible model to represent MSE and the interactions possible in them, which acts a semantic interoperability platform by enabling different applications to share and reuse their respective knowledge. This supports interoperability between separately developed applications. (ii) It implements a set of components to help social application developers manage their MSE by allowing social information storage and retrieval, and provides a set of support tools to generate code based on the application data model. (iii) It supports access control to the user's social data based on socially aware policies, i.e., it allows each user to customize access to his/her social data based on social information itself, as well as context. (iv) It provides a versatile support for decentralized mobile social applications, which enables execution on both computers and smart phones with minimal configuration effort, and seamless communication over heterogenous wireless networks, allowing users to maintain social data local to their device(s).

The rest of the paper is organized as follows. In Section 2, we highlight the differences of our work with respect to existing work in the domain. Our contributions are discussed in detail in Section 3, where we discuss the architecture of Yarta. Section 4 provides the implementation details of our middleware, as well as an extensive evaluation of the prototypes in terms of performance, scalability, as well as ease of use in using Yarta to develop applications. Section 5 concludes with a sketch of our planned research in the near future.

## 2 Related Work

Current MSAs, such as applications supporting the dissemination of content updates (e.g., news or traffic information) over a mobile social network [12], or exploiting mobile social networking to enhance group communication [9], are often designed from scratch by embedding into the application logic all MSE management functionalities and providing application-specific data representation models. In this section we focus on existing platforms and middleware architectures for supporting mobile social applications. An overview of related research regarding MSAs and MSE modeling can be found in [20].

Some authors have recognized the need to externalize social management functionalities [17], [21], but to the best of our knowledge, only a few middleware frameworks to support MSAs have been proposed. Table 1 summarizes the main contributions and drawbacks of existing solutions with respect to the challenges discussed in Section 1. The MobiSoc [10] and MobiClique [16] middleware provide simple data models compared to Yarta MSE model, while the IYOUIT application offers a wide set of concepts to model users' activities and interactions [3]. IYOUIT cannot be considered a middleware since it is provided as a stand-alone application, while MobiSoc and MobiClique both provide open APIs.

**Table 1.** Comparison of middleware for supporting mobile social applications

Middleware	MSE Model	Privacy & Access Control	MSE Mgmt Features	Decentralized Architecture	Reusability
MobiSoc	People, Place People-to-People, People-to-Place	Authentication, Confidentiality (Encryption)	Social Data Inference, Event Mgmt.	No	Open APIs
MobiClique	User Profile (specific format)	No	Proximity-based Social Interaction Data Extraction	Yes (Opportunistic Networks)	Open APIs
Middleware PSC	Augmented FOAF (Tasks, Preferences)	No	User's Task Matching	Optional	N/A
IYOUIT	Location, Experience, Pictures, Interests, Buddies, ...	Permissions (identity/group)	Social Data Inference, Friends Network	Distributed	Proprietary
Prometheus	People, type of relation, strength	Privacy (Encryption) Access Control, Trust	Social Inference over Multiple Sources	Yes	Open APIs
PrPI	Multi-application	Authentication (OpenID)	Access to social data, query	Decentralized	Open API, SocialLite Language
Yarta	Agent, Event, Place Content, Topic (extensible)	AC Policies, Authentication, Confidentiality	MSE Creation, Update, Exchange, & Extraction	Multi-radio Multi-platform (iBICOOP)	Open APIs (LGPL)

The middleware for Pervasive Social Computing proposed in [1] adopts a similar model to Yarta (RDF-based), which however lacks in generality. In addition, the middleware is not available for reuse. Also worth mentioning is Motorola's soon-to-be-discontinued MOTOBLUR<sup>3</sup> UI, which aggregates updates from a user's social networks and allows posting in multiple places. The PrPI architecture, targeted to decentralized environments, allows users to access (and share) data stored by different social applications by a special purpose query language [18]. Prometheus is a P2P architecture that collects and manages social information from multiple sources and implements a set of social inference functions [15].

Yarta differs from all previous approaches since it allows the exchange of social graphs between users, and between applications, in an interoperable format, which also allows reasoning. Additionally, to the best of our knowledge, Yarta is the only MSE middleware that can fully execute on mobile phones. Finally, its access control model is socially aware and takes advantage of semantic reasoning.

### 3 The Yarta Middleware

To address the challenges discussed in Section 1, we have developed the Yarta middleware architecture, consisting of two layers (see Fig. 1): the *MSE Management Middleware* layer, managing and allowing access to MSE data, and the *Mobile Middleware* layer, handling low level communication/coordination issues.

#### 3.1 Managing Knowledge in Mobile Social Ecosystems

Yarta is based on an expressive and extensible model to represent MSE, which acts a semantic interoperability platform by enabling different applications to share and reuse their respective knowledge. We first describe the data model, and then present features offered by the middleware component in charge of managing the user's Knowledge Base.

<sup>3</sup> <http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/MOTOBLUR/Meet-MOTOBLUR>

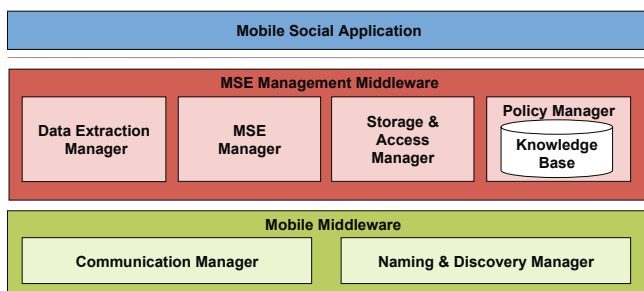


Fig. 1. Yarta Middleware Architecture

**MSE Representational Model.** Our model is based on the representational model in [20], to which we have added more details, such as a name, a latitude and a longitude value for each *Place*, as suggested by most existing ge-positioning standards<sup>4</sup>, and unique IDs for resources<sup>5</sup>. Note that this model re-uses concepts defined in existing social information and content ontologies such as Friend-of-a-Friend (FOAF)<sup>7</sup> and the Dublin Core<sup>6</sup>.

Both the base model defined in [20] and our augmented model discussed above are represented using the Resource Description Framework (RDF)<sup>8</sup>, a base Semantic Web standard. Because all mobile social applications share a common data model, they rely on a shared platform of common meaning, which they can further extend based on specific requirements: by means of automated reasoning, classes and properties defined in application-specific extensions are put in clear semantic relation with base classes, thus enabling data interoperability. We provide a detailed example of this in Section 4.2.

**Accessing and Processing Social Data.** Data represented according to the above model are connected to form a uniform graph of social information. This is well suited to a scenario where each user owns his data locally, and autonomously manages his graph by possibly adding to it portions of graphs provided by other users, as well as by other sources of social information (see Section 3.2). The social graph is managed by the *Knowledge Base* (KB) middleware component.

The KB offers both low-level (RDF-oriented) as well as high-level APIs to access, update and remove social data. These functionalities are provided independently of the specific implementation chosen for semantic data management. The KB is also able to handle the merging of MSE graphs coming from different users. Finally, the KB is wrapped by a *Policy Manager* to ensure access control enforcement according to the policies defined in the system, as explained later.

<sup>4</sup> [http://www.w3.org/2003/01/geo/wgs84\\_pos#](http://www.w3.org/2003/01/geo/wgs84_pos#)

<sup>5</sup> Due to lack of space we refer the reader to [20] for the graph of first-class entities and relationships.

<sup>6</sup> <http://www.w3.org/TR/rdf-primer/>

### 3.2 Providing MSE Management Functionalities

Yarta supports mobile application developers along three directions:

**Knowledge Abstractions for Application Developers.** Yarta provides an access interface to the KB from the user’s perspective. Specifically, it allows the user application to add and retrieve information to/from the KB by means of high level queries hiding the details of the internal RDF representation (e.g., nodes or triples). It also allows the execution of remote queries, performed over the KB of another user. This is done via a dedicated component, namely the *Storage & Access Manager*. In particular, the task of providing a high-level abstraction of the knowledgebase to the application developers is divided into two parts. Firstly, the developer is provided a `ResourceFactory` class which he can use to create new instances of resources. These objects in turn provide the basic getter and setter methods of a Java Bean for each of its properties, as well as properties inherited from its superclasses.

Secondly, the Storage and Access Manager also provides the users with methods to add/delete/query the relationships between resources as defined by the model (e.g. `addMember(Agent, Group)`, `getMembers(Group)`, `getMembers_inverse(Agent)`, and `isMember(Agent, Group)`). Additionally, the `StorageAccessManager` class also provides an `executeRemoteQuery(remotePeer, query)` method which can be used to execute a subset of these queries on the KBs of a user’s peers, and add the information returned to the user’s KB in accordance with the access control policies in place.

**Automatic Generation of API to Access KB.** We have designed a tool to alleviate the burden of developers who want to extend the core model of Yarta to develop their own applications. Specifically, this tool takes an RDF model representing the types of resources in the application and the relationships between them, and generates source files which provide an object-oriented API over the knowledgebase. The facility is akin to those provided by toolkits used by developers interacting with database systems[11]. Note that an essential feature of an API generator for Yarta is the support for *multiple inheritance*, a feature provided by RDF, but not natively supported by languages such as Java.

**Extracting Data from Social Sensors.** We have designed the *Data Extraction Manager* to populate the user’s KB by extracting data from two distinct types of sources. The first already contain social links such as “friendship” in addition to general information, while the second do not contain social links, but may contain information which can be correlated to infer social links. For the former, adapters can be written in their API to import their data into Yarta; while for the latter, we need to employ inference algorithms to correlate data and guess/recommend social links. The structure of the Data Extraction Manager is modular to allow for plug-and-play behavior of adapters.

### 3.3 Controlling Access to MSE Knowledge

Yarta provides a flexible and powerful support for access control, which can be fine-tuned based on the social preferences of the user. In particular, it adopts semantic policies to define access directives. Policies are completely decoupled from both the application logic and the KB management, to allow fine-grained and customizable security behavior. Policies allow read/add/remove actions on triples or graphs (i.e., sets of triples) and are modeled based on the socially aware policy model presented in [19]. As a key feature, they define access rules to data based on social information. Policies are currently represented as a combination of SPARQL queries and RDF statements. For the sake of conciseness we do not describe the policy model in detail, but refer the reader to [19].

The Yarta middleware includes a dedicated component for the definition, management, evaluation and enforcement of access control policies over the KB, called *Policy Manager*. The Policy Manager intercepts any tentative access action on the KB, and performs reasoning on defined policies and the access request's context (e.g., relation with the requester, properties of resource, etc.) to determine whether the action is permitted.

### 3.4 Supporting Decentralized and Heterogeneous Environments

Yarta is designed to execute in ubiquitous environments, characterized by a high degree of heterogeneity in both connectivity options and hardware platform. In addition, it does not assume any centralized server to collect and manage the user's data, nor to perform any other social functionality, as detailed below.

**Communicating over Heterogeneous Networks.** Yarta is supported by a multi-platform communication layer that offers both synchronous and asynchronous messaging support over multi-radio links, and supports data transfer over heterogeneous network interfaces and connecting technologies, even in case of temporary disconnections due to user mobility. In addition, it provides support for network-agnostic service/device naming and discovery. Yarta also implements base security features via authentication and confidentiality mechanisms.

**Execution on Multiple Mobile Platforms.** Yarta is developed in Java to take maximum advantage of portability across mobile platforms. Yarta can execute on different nodes: smart phones running Android, and laptops/workstations with different operating systems thanks to the language portability. This applies to all middleware components and actually allows the deployment of the whole platform on resource-constrained devices, without the need for an external proxy handling semantic processing.

## 4 Implementation and Evaluation

### 4.1 Implementation Details

The Yarta middleware prototype [22] is written in Java2 SE and has been deployed both on laptops running Windows/Mac OS, and on smart phones running

Android. In the laptop prototype, both the Knowledge Base and the Policy Manager rely on capabilities offered by the Jena Semantic Web Framework [13]. The KB currently uses the filesystem as a backing store. For the Android prototype we exploit Androjena<sup>7</sup>, an Android-compatible port of the Jena framework.

As a first implementation of the Data Extraction Manager’s social sensors, we wrote adapters for Facebook and LinkedIn using their native APIs. For the data sources which are not intrinsically social, we implemented adapters which used the data stored in the user’s phone contacts, and correlated it with his call logs and SMS logs to draw some basic inference about the contacts whom a user “knows”. This work is at an early stage, but the modular architecture of the data extraction manager is found to be suitable for easily writing more adapters, or testing better inference algorithms. To help developers in extending the core Yarta model to for their application, we have implemented the API generator discussed in Section 3 by extending the Jastor<sup>8</sup> toolkit. For Android phones, we also provided a wrapper of the storage and access manager in the form of a `ContentProvider`, the standard way for Android applications to access data.

Finally, we exploit the iBICOOP middleware [2] as part of our Communication Manager and Naming/Discovery Manager to provide discovery and messaging abilities. Yarta services residing on mobile devices are accessed through their iBICOOP URL (iBIURL), created by combining the `userID` and `deviceURI` from our model with the Yarta application name and the specific service needed.

## 4.2 System Evaluation

To evaluate Yarta, we collected data on both the scalability of the core features provided by Yarta, as well as the development effort involved in building new applications on top of it.

**Performance and Scalability Evaluation.** To evaluate the scalability of Yarta, we profiled its behavior during several operations, including those for adding a new person to the KB, adding a KB from a file, retrieving a person and his acquaintances from the KB, and performing operations on a remote node.

For each of the above methods, we measured the time taken for their execution on the Google Nexus One mobile phones running Android 2.2 OS, with a 1 GHz processor and 512 MB of RAM. We used KB sizes ranging from 1 to 1001 entries in increments of 100, with 5 applicable access control policies in place, and averaged the times taken in performing the operations for 10 runs.

We present the observations from our experiments in Figures 2 and 3. Figure 2(a) illustrates the time taken for adding information in the KB from a file, and the time taken for reading one person from that KB after that. Note that although the times increase with the size of the KB, the time taken for the interactive `getPersonByUID` method is well within the 5 second interactive UI response time limit used by the Android OS. The same can be seen for the interactive `addPerson` method profiled in Figure 2(b). Further, Figure 3(a) presents

<sup>7</sup> <http://code.google.com/p/androjena/>

<sup>8</sup> <http://jastor.sourceforge.net>

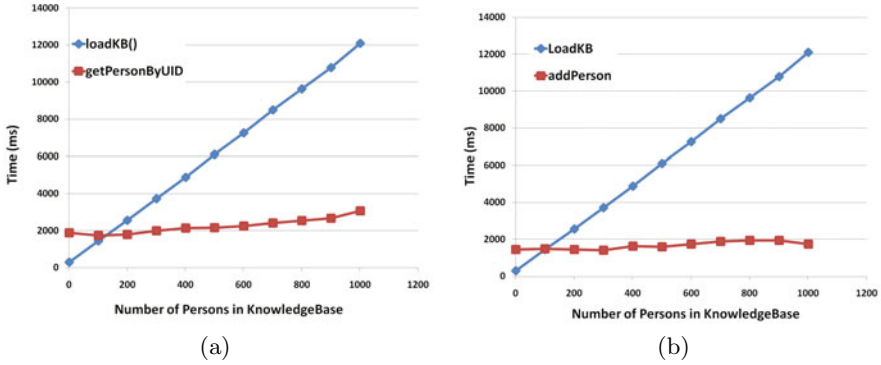


Fig. 2. Time taken for loadKB and then a) getPerson b) addPerson

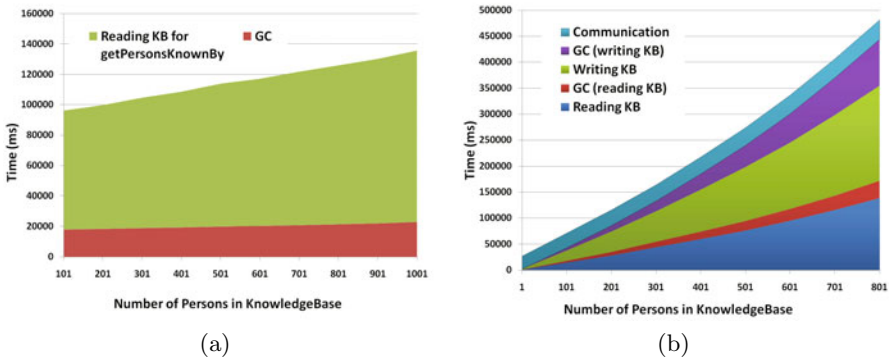


Fig. 3. Time taken for a) getPersonsKnownBy and b) remoteGetAllPersons

the amount of time taken for getting the list of persons known by someone (set to return 50 persons in the KB), which was seen to take inordinately long times for the user to wait. Since Android allows lists to be populated in an incremental manner as data flows in, this may be used to mitigate the slow response time of this operation. Finally, Figure 3(b) provides the details of the time taken in the various steps of getting the list of all the persons in the KB of another peer. We have noted the time taken for reading the remote KB and writing the information in the local KB, with the latter being higher since Jena performs duplication checks at write time. Note also the time taken by Android in garbage collection during the read and write operations. After inspecting and optimizing our code, we believe that the next step would be to improve the Androjena library so that it uses memory more efficiently. The other operations were also seen to perform within acceptable time bounds with similar profiles. On the laptop, we observed similar trends in the times taken, but the actual times taken were much less, owing to better hardware.

**Developing mobile social applications on top of Yarta.** To showcase the real-world applicability of our approach and the extensible nature of our middleware discussed in Section 3, we developed two proof-of-concept applications that allow mobile users to perform various social activities in different scenarios discussed in Section 1 – a *Conference Mate* app to assist the attendees of a conference, and a *FIFA* app to allow the fans attending a football match.

*Extending the Data Model.* For both applications we extended the Yarta data model to include concepts specific of each scenario. Extension proceeds by subclassing RDF classes/properties and possibly importing other (portions of) ontologies, provided that they are compliant with the base model. Extensions included subclasses of Event (*Talk*, *Coffee Break* and *Meeting*) for the Conference Mate app, and *Match* as a subclass of Event for the FIFA app. For the sake of brevity, we do not mention all the defined classes and properties here. After defining the data model, we created access control policies for the MSE data, including complex application-specific policies such as “any friend (person I know) that supports my team is allowed to read the list of my friends who also support the same team, as well as their affiliation to my fan club” for the FIFA app.

*Writing the Application Code.* The first step toward the development of an application is the extension of the Storage and Access Manager to incorporate the extensions made to the model. For this purpose, we used the automatic code generation tool described in Section 3.2. This greatly helped reduce the programmer’s burden, since the  $\sim 5000$  lines of code for the new API for the FIFA app were automatically generated, for example. Overall, this auto-generated code constituted  $\sim 70\%$  of the total code of the apps, with the developer-written code mostly implementing a user interface over the API provided by Yarta.

In summary, our evaluations show that Yarta performs well in terms of efficiency and expressiveness, the two main desirable properties of middleware.

## 5 Conclusions and Future Work

In this paper, we presented Yarta, a novel middleware for supporting complex mobile social ecosystems of the not-so-distant future. Our middleware allows knowledge exchange between users and between applications, and provides flexible policies controlling access to MSE data based on users’ social preferences. Yarta implements a set of components that allow social information storage and retrieval, automatic generation of code for mobile social applications, and extraction of social data from available social sensors. It can execute on smart phones and laptops and is able to communicate over heterogeneous wireless networks. We demonstrated the efficacy and usability of our middleware by providing an extensive evaluation of the middleware and two prototype applications, running on Android phones, that we developed on top of it. We are currently working on the Yarta middleware along several directions, including developing more advanced algorithms to extract MSE from location/context data sets, providing the KB a database backing store, and improving performance on mobile phones.



## References

1. Ben Mokhtar, S., Capra, L.: From pervasive to social computing: algorithms and deployments. In: ICPS 2009: Proceedings of the 2009 International Conference on Pervasive Services. ACM, New York (2009)
2. Bannaceur, A., Singh, P., Raverdy, P.G., Issarny, V.: The ibicoop middleware: Enablers and services for emerging pervasive computing environments. In: PerCom Workshops, pp. 1–6. IEEE Computer Society, Los Alamitos (2009)
3. Boehm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, M., Wibbels, M.: Introducing IYOUIT. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 804–817. Springer, Heidelberg (2008)
4. Churchill, E.F., Halverson, C.A.: Guest editors' introduction: Social networks and social networking. IEEE Internet Computing (2005)
5. The Diaspora Project, <http://www.joindiaspora.com/> (last visited: May 2010)
6. Dublin Core metadata element set, version 1.1, <http://www.dublincore.org/documents/dces/> (last visited: May 2010)
7. Friend of a Friend, <http://www.foaf-project.org/> (last visited: March 2010)
8. Foth, M.: Facilitating social networking in inner-city neighborhoods. IEEE Computer 39(9), 44–50 (2006)
9. Grob, R., Kuhn, M., Wattenhofer, R., Wirz, M.: Cluestr: mobile social networking for enhanced group communication. In: GROUP 2009: Proceedings of the ACM 2009 International Conference on Supporting Group Work, pp. 81–90 (2009)
10. Gupta, A., Kalra, A., Boston, D., Borcea, C.: MobiSoC: a middleware for mobile social computing applications. Mob. Netw. Appl (2009)
11. Hibernate, relational persistence for java and .net, <http://www.hibernate.org/>
12. Ioannidis, S., Chaintreau, A., Massoulié, L.: Distributing content updates over a mobile social network. ACM SIGMOBILE Mobile Computing and Communications Review 13(1), 44–47 (2009), <http://dx.doi.org/10.1145/1558590.1558599>
13. Jena, <http://www.jena.sourceforge.net/> (last visited: May 2010)
14. Jones, Q., Grandhi, S.A.: P3 systems: Putting the place back into social networks. IEEE Internet Computing 9(5), 38–46 (2005)
15. Kourtellis, N., Finnis, J., Anderson, P., Blackburn, J., Borcea, C., Iamnitchi, A.: Prometheus: User-controlled P2P social data management for socially-aware applications. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 212–231. Springer, Heidelberg (2010)
16. Pietiläinen, A., Oliver, E., LeBrun, J., Varghese, G., Diot, C.: MobiClique: middleware for mobile social networking. In: Proceedings of the 2nd ACM Workshop on Online Social Networks, pp. 49–54. ACM, New York (2009)
17. Rana, J., Kristiansson, J., Hallberg, J., Synnes, K.: An architecture for mobile social networking applications. In: First International Conference on Computational Intelligence, Communication Systems and Networks, CICSYN 2009, pp. 241–246 (July 2009)
18. Seong, S.W., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Teh, S.K., Chu, R., Dodson, B., Lam, M.S.: Prpl: a decentralized social networking infrastructure. In: MCS 2010: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services, pp. 1–8. ACM, New York (2010)

19. Toninelli, A., Montanari, R., Lassila, O., Khushraj, D.: What's on users' minds? toward a usable smart phone security model. *IEEE Pervasive Computing* 8(2), 32–39 (2009)
20. Toninelli, A., Pathak, A., Seyedi, A., Cardoso, R.S., Issarny, V.: Middleware support for mse management. In: *Proceedings of the 2nd IEEE International Workshop on Middleware Engineering, to be held with COMPSAC 2010* (2010)
21. Tran, M., Han, J., Colman, A.: Social context: Supporting interaction awareness in ubiquitous environments. In: *6th Annual International on Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous 2009*, pp. 1–10 (July 2009)
22. Yarta, <https://www.gforge.inria.fr/projects/yarta/> (last visited: May 2010)

# A Coordination Middleware for Orchestrating Heterogeneous Distributed Systems

Nikolaos Georgantas, Mohammad Ashiqur Rahaman, Hamid Ameziani,  
Animesh Pathak, and Valérie Issarny

INRIA Paris-Rocquencourt, France  
firstname.lastname@inria.fr

**Abstract.** Integration of heterogeneous distributed systems becomes particularly challenging when these systems have diverse coordination models (e.g., client/server, publish/subscribe, tuple space). In this paper, we introduce a system integration solution based on orchestration workflow and a high-level data-driven coordination abstraction enabling application workflows that are agnostic to the underlying middleware platforms and associated coordination models of the constituent systems. Our solution features an extensible generic coordination middleware, which enables middleware designers to easily incorporate support for new middleware platforms and facilitates application designers in designing complex applications.

## 1 Introduction

In the near future, complex distributed applications will be to a large extent based on the integration of extremely heterogeneous systems, such as lightweight embedded systems (e.g., sensors, actuators and networks of them), mobile systems (e.g., smartphone applications), and resource-rich IT systems (e.g., systems hosted on enterprise servers and Grid infrastructures). These heterogeneous system domains have developed individually, with significant advancements in terms of coordination models (CMs), communication protocols, data representation models, as well as related middleware platforms that incorporate the former thus providing integral support to applications. In particular with regard to middleware-supported coordination, the client/server (CS), publish/subscribe (PS), and tuple space (TS) models are among the most widely employed ones, with numerous related middleware platforms, such as: Web Services, Java RMI for CS; JMS, SIENA for PS [1,2]; and JavaSpaces, Lime, DART for TS [3,4,5].

In the following, we outline a representative application scenario, where a complex distributed application needs to be devised by integrating heterogeneous systems that interact with varying CMs. This scenario concerns Search and Rescue (S&R) operations after a disaster, such as a flood or an earthquake.

S&R operations are carried out in a highly hazardous environment with large numbers of injured parties entrapped in scattered places. In such situations, personnel from multiple agencies (i.e., fire-fighters, police,

Red Cross) must coordinate. To detect survivors, sensor nodes are installed at various places of the hazardous area. Upon installation/move, such nodes communicate their GPS coordinates. S&R personnel also notify automatically at short intervals of their current positions via their PDAs. Upon sensing some life sign, sensor nodes send out notifications. At the same time, nearby light-emitting actuators start lighting the place to facilitate the rescuing effort. Sensors, PDAs, and actuators interact among them and with external actors via a TS. The above positioning data/life sign notifications are sent periodically/instantly via a CS invocation to a planning service that calculates distances and paths and provides/recommends at real time the current/optimal deployment of rescue forces. This output is then notified via a PS system to the coordinator of the operation on her smart phone and also to a number of control/monitoring centers. The coordinator may approve and command appropriate personnel via the PS system to rush into the spot. Commands reach the personnel via their TS-enabled PDAs.

To enable such a scenario, the heterogeneity between the involved system domains needs to be tackled. Existing cross-domain interoperability efforts are based on, e.g., bridging communication protocols [6], wrapping systems behind standard technology interfaces [7], and/or providing common API abstractions [8,9,10,11]. However, such efforts commonly cover part of the heterogeneity issues (regarding coordination, communication, data) and are applicable to specific cases. In particular, existing solutions do not or only poorly address CM interoperability.

In this paper [4], we propose a model-based system integration approach that can deal with diverse existing systems, focusing in particular on integrating their heterogeneous CMs. Firstly, our solution features workflow-based orchestration of the constituent systems. Orchestration workflow is a well-established paradigm for composing different systems under the control of a central coordinating entity. A number of models, languages and associated support platforms have been proposed for designing and executing orchestrations, however, focusing almost exclusively on service-oriented systems, i.e., assuming CS coordination. Among them, BPEL is the most widely used solution, a standard for Web Services orchestration [2]. Envisaging heterogeneous orchestrations, we extend the workflow-based orchestration model to cover diverse CMs. However, this results in additional complexity for the application designer, who is required to deal in her workflow with all the heterogeneous CMs of the constituent systems. Thus, secondly, our solution introduces a single higher-level CM to abstract from the underlying CMs of the constituent systems. We call the former *application CM* and the latter *middleware CMs*, thus introducing a decoupling between the abstracted CM to be employed in application workflow design and the native CMs of the constituent systems to be employed in the execution of the application

<sup>1</sup> This work has been partially supported by the ITEMIS project (<http://itemis-anr.org/>) funded by the French National Research Agency (ANR).

<sup>2</sup> <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>

workflow. To elicit a unique application coordination abstraction, we observe that conveying application data is a common denominator of all middleware CMs. This leads us to propose data-driven application-level coordination. We specifically introduce *Global Data Space (GDS)* as part of our workflow model, a special TS abstracting interactions between workflows and constituent systems. Inherent properties of the TS model, such as time and space decoupling among communication entities [4], make it sufficiently generic for abstracting complex distributed applications that integrate heterogeneous systems.

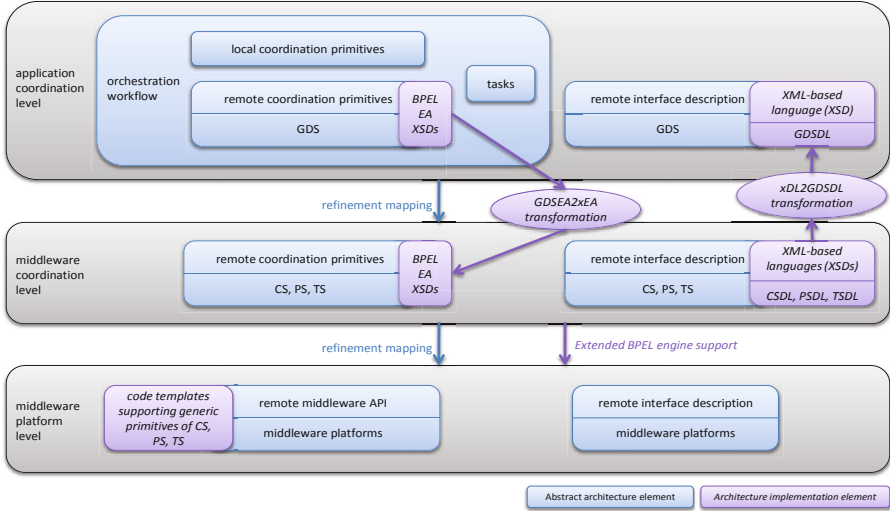
Our systematic abstraction approach is carried out in two stages. First, a middleware platform is abstracted under a corresponding middleware CM among the three principal ones, i.e., CS, PS and TS. To this aim, we elicit generic coordination primitives for each model comprehensively covering their essential semantics, after having thoroughly surveyed the related literature. Then, these three models are abstracted further into the unique application CM. We introduce high-level generic coordination primitives for the GDS model (derived from the TS primitives) and mappings between them and the primitives of each one of CS, PS and TS. Our two-step abstraction and decoupling between application- and middleware-level coordination enable orchestrations that are agnostic to the underlying middleware platforms of the heterogeneous constituent systems.

The above concepts are incorporated into an abstract coordination middleware architecture (Section 2), which we implement into a prototype extensible coordination middleware by building upon BPEL and its extensibility support mechanism (Section 3). We further explicitly identify the roles of the application and middleware designers and provide them with design-time support: enabling the former to easily design application workflows integrating heterogeneous constituent systems; and the latter to rapidly add support for new middleware platforms into the prototype coordination middleware. We demonstrate the applicability of our approach by implementing the representative scenario introduced above. Based on this implementation, we evaluate our approach by applying design time metrics for measuring the support offered to the application and middleware designers (Section 4). Our solution considerably reduces the efforts of both designers in dealing with multiple heterogeneous middleware platforms. We complement this paper with a comparison of our approach with related work (Section 5), and conclude by discussing future work (Section 6).

## 2 Abstract Coordination Middleware Architecture

We introduce an abstract model-based architecture that embeds our application and middleware CMs, their coordination primitives, and the refinement mappings between them. The abstract architecture is depicted in Fig. 1, together with the architecture implementation elements, which will be introduced in Section 3. The architecture is structured in three abstraction levels. We present in the following each one of these levels, and briefly discuss the elicited primitives and their mappings.

**Application Coordination Level.** The application coordination level provides a high-level abstraction of a complex distributed application. Such an application



**Fig. 1.** Abstract Coordination Middleware Architecture and its Implementation

is designed as an orchestration workflow constituted by tasks. Tasks employ local coordination primitives for control passing (e.g., implementing constructs like sequence, parallel execution, etc.), and for data passing (e.g., via global variables accessible to all tasks) among them. Some of the tasks employ remote coordination primitives for control and data passing from/to external systems that are integrated by the workflow. Remote coordination is based on the GDS model. Thus, application workflows can be designed based on the GDS CM for interacting with external constituent systems. We point out that given the GDS abstraction, the application workflow designer needs not deal with the heterogeneous CMs and middleware platforms of the constituent systems. To properly interact with such systems, the workflow designer needs to have their interface descriptions, which should also be expressed based on the GDS model.

**Middleware Coordination Level.** The middleware coordination level offers its proper remote coordination primitives for interaction with the constituent systems, as well as the interface descriptions of such systems, to which the interaction should conform. These coordination primitives and interface descriptions comply with the CS, PS and TS CMs. Hence, the interaction with the constituent systems is performed at this level based on the systems’ native CMs. The middleware coordination level provides a refinement to the application coordination level with respect to the primitives and interfaces. Thus, there is a refinement mapping between these elements and the respective GDS-based elements. The rest of the elements of the application coordination level (local coordination primitives, tasks not dedicated to remote interaction) remain intact when this refinement is applied.

**Middleware Platform Level.** The middleware platform level enables the final refinement mapping of remote coordination primitives and interface descriptions to the ones provided by the concrete middleware platforms of the constituent systems. This means that interaction with the constituent systems is performed at this level via the APIs of these middleware platforms. Hence, these platforms should be incorporated by our coordination middleware implementation. In the same way, interface descriptions of constituent systems comply to their middleware platforms APIs.

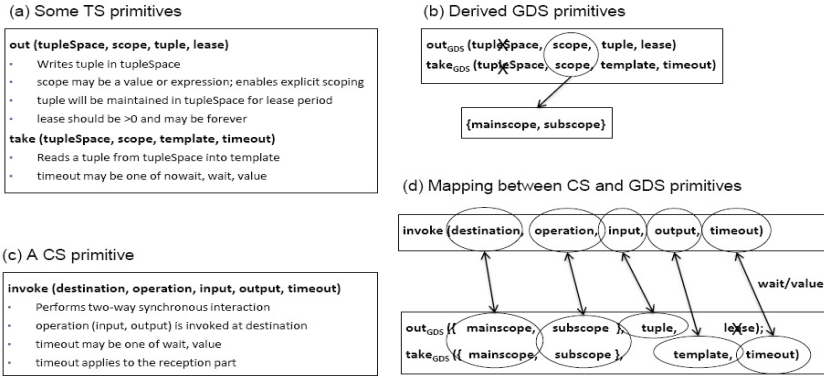
**Coordination Primitives.** As already discussed, we have elicited generic primitives and their arguments for the CS, PS and TS models after extensively surveying these models and related middleware platforms in the literature. We then derived primitives for the GDS model from the TS primitives. Due to space restrictions, we cannot present the complete primitives and their mappings; we only briefly discuss the primitives and illustrate their mapping with an example.

CS primitives are inspired from BPEL and Web Services interaction primitives, covering one-way asynchronous and two-way synchronous interaction in both directions between client and server. For PS, we cover queue-, topic- and content-based systems [12], integrating the diverse event types under a generic *filter* parameter. For TS, we integrate support for asynchronous notification [3] and explicit scoping [5]. GDS primitives are special TS primitives.

Fig. 2 depicts two TS primitives, the derived GDS primitives, and how a CS primitive is mapped onto the ordered sequence of these GDS primitives. More specifically, the two-way interaction of the CS *invoke* primitive is mapped onto a sequence of *out<sub>GDS</sub>* and *take<sub>GDS</sub>* primitives. While most explanations are given in the figure, we point out the introduction of the explicit scoping parameters *mainscope* and *subscope* in the GDS primitives. These two parameters enable restricting the scope of the entities that share GDS data. We map the *destination* and *operation* parameters of *invoke* to *mainscope* and *subscope*; thus, the *input* and *output* data carried by *invoke* will be accessed only by the single CS destination peer.

### 3 Coordination Middleware Implementation

We present in this section our implementation of the abstract coordination middleware architecture, building on BPEL. BPEL already implements some of the elements of the abstract architecture. It offers complete language support (tasks, local coordination primitives) for developing workflows and executing them via its engine. However, it provides limited remote coordination primitives and related remote interface description (based on WSDL), as well as limited middleware platform level support, targeting only Web Services. Nevertheless, BPEL allows enhancing the base language with *extension activities (EAs)* and associated user-defined refinement mappings at the middleware platform level. We exploit this powerful feature to implement the novel elements of the abstract architecture.



**Fig. 2.** An example of coordination primitives and their mappings

We identify three roles related to our coordination middleware. The coordination middleware designer (this is us) develops the *core coordination middleware*. A middleware platform designer can incorporate support for a new middleware platform into the core middleware. An application workflow designer can design a workflow that integrates a number of heterogeneous constituent systems.

In the following, we present the core middleware implementation and its integrated support for the middleware platform (hereafter, middleware) and application designers. We demonstrate the applicability of our approach by implementing the application scenario outlined in the Introduction. In particular, our scenario implementation, depicted in Fig. 3, integrates: (1) sensors, actuators and personnel equipment communicating over a DART TS [5]; (2) the planning service implemented as an Apache Axis2 Web Service[3]; (3) a JMS PS system based on Apache ActiveMQ[4] that the coordinator of the operation uses to receive recommendations and to send commands; and (4) the application workflow developed and running on the Apache ODE BPEL workflow engine[5].

**Core Middleware.** The core coordination middleware provides generic support for application workflow development based on the GDS model, and for the CS, PS and TS models at middleware coordination level. The core middleware implementation elements are depicted in Fig. 1. We introduce new BPEL EAs (in the form of XML Schema or XSD) representing the required application- and middleware-level remote coordination primitives. At platform level, we introduce support for the middleware-level EAs. In particular, we provide code templates (to be filled in by the middleware designers) for enabling later linking the CS, PS and TS generic primitives to middleware platforms that will be incorporated into the core middleware. Regarding interface descriptions at middleware level, we introduce three XSD-based description languages (DLs), inspired from WSDL:

<sup>3</sup> <http://axis.apache.org/axis2/java/core/>  
<sup>4</sup> <http://activemq.apache.org/>  
<sup>5</sup> <http://ode.apache.org/>



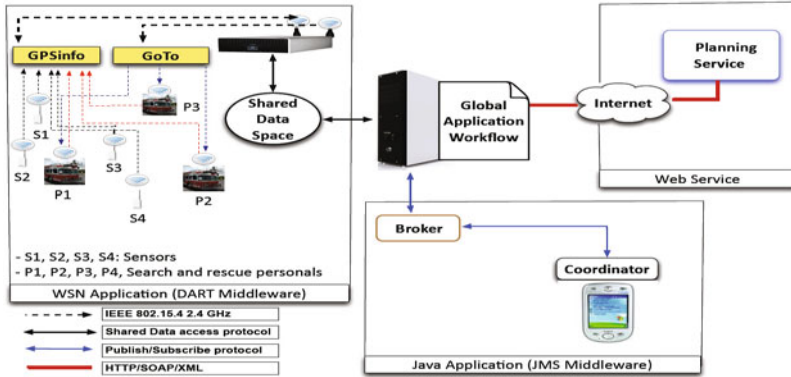


Fig. 3. Search and Rescue Operations

CSDL, PSDL and TSDL, conforming to the respective CMs and their generic primitives. For interface descriptions at application level, we introduce in the same way one more language based on the GDS model: GDSDL which will be used by the workflow designers. Finally, to facilitate the task of the workflow designers, we propose two XSLT-based transformations [13], which implement the refinement mappings between the application and the middleware levels: (1)  $xDL2GDSDL$ , transformation of middleware-level interfaces to application-level interfaces, and (2)  $GDSEA2xEA$ , transformation of application-level EAs to middleware-level EAs, where  $x=CS/PS/TS$ .

**Integrating a New Middleware Platform.** The core middleware provides generic support for the CS, PS and TS models as well as for the GDS model, in terms of primitives, interfaces, code templates and XSLT-based transformations. Based on this, on-demand integration of a new middleware platform is basically a set of design-time activities as follows: (1) Embedding platform-dependent code into the code templates so as to employ the platform API. (2) Refining middleware-level EAs and the interface xDL; this is about type refinement, so that generic data types of the core middleware are customized to the specific middleware platform. (3) Refining application-level EAs and the GDSDL; this is the same type refinement as in the previous step. In case the new middleware platform introduces features not covered by the core middleware, the core middleware can be extended to embed these features. We incorporated three platforms into the core middleware: Apache Axis2 (CS middleware), ActiveMQ JMS (PS middleware) and DART (TS middleware).

**Designing an Application Workflow.** Based on the generic support of the core coordination middleware and the on-demand integration of middleware platforms, the application workflow designer can design a new workflow, using her sole knowledge of the GDS CM. In particular, she can design a data-oriented workflow of heterogeneous constituent systems independently of their CMs. The

specific design-time activities are as follows: (1) Transforming native interfaces of the constituent systems to corresponding xDL description, and then to GDSDL description by employing the xDL2GDSDL transformation. (2) Designing the GDS-based workflow by writing tasks that employ appropriate GDS EAs. (3) Refining the GDS-based workflow into an executable workflow by using the GDSEA2xEA transformation. (4) Finally, deploying and executing the workflow on a BPEL support platform, e.g., Apache ODE.

## 4 Middleware Evaluation

We have evaluated our middleware by applying design time metrics for measuring the support offered to middleware and application designers, as introduced in the previous section. Due to space limitations, we present herein our evaluation results only for the former support. More specifically, we measure the effort of the coordination middleware designer for implementing the core middleware, and the effort of middleware platform designers for integrating two new middleware platforms into the core middleware, namely the JMS and DART platforms of our application scenario.

**Coordination Middleware Designer.** Table 1 summarizes her effort in providing support for the PS and TS CMs at middleware coordination level, in terms of implemented numbers of: (1) BPEL EAs representing coordination primitives and their arguments, (2) XSD elements of xDL interface description languages, (3) XSLT expressions of xDL2GDSDL transformations, and (4) Lines of code of code templates at middleware platform level.

**Table 1.** Effort of the coordination middleware designer

BPEL EAs	PS		TS	
	Primitives	Arguments	Primitives	Arguments
	3	7	5	9
xDLs (XSD elements)	PSDL		TSDL	
	11		14	
Transformations (XSLT expr)	PSDL2GDSDL		TSDL2GDSDL	
	42		63	
Code templates (LOC)	1002		1912	

**Middleware Platform Designer.** Table 2 shows the efforts of the JMS (a PS middleware) and DART (a TS middleware) designers for integrating their platforms into the core middleware, in terms of implemented numbers of: (1) New BPEL EAs representing new coordination primitives and/or new arguments, in case they are not covered by the already implemented generic core middleware primitives, (2) New XSD elements of xDLs, as for data type refinement, (3) New XSLT expressions of xDL2GDSDL transformations, in case this is needed due to xDL refinement, and (4) New lines of code incorporated into code templates

**Table 2.** Effort of the middleware platform designers

	JMS		DART	
	New primitives	New arguments	New primitives	New arguments
BPEL EAs	0 (0%)	4 (36%)	0 (0%)	2 (18%)
xDLs (New XSD elements)	PSDL 6 (35%)		TSDL 2 (12.5%)	
Transformations (New XSLT expr)	PSDL2GDSDDL 0 (0%)		TSDL2GDSDDL 0 (0%)	
Code templates (New LOC)	508 (34%)		311 (14%)	

at middleware platform level. Besides absolute values, Table 2 also shows the ratio of the designers' actual effort over the total effort they would have had to put in case the core middleware had not been already available. The outcome ratio numbers point out the significant support offered to the middleware platform designers, resulting in considerable easiness for integrating new middleware platforms and related high extensibility of the coordination middleware.

## 5 Related Work

Distributed system interoperability approaches at the middleware level are classically based on bridging communication protocols, wrapping systems behind standard technology interfaces, and/or providing common API abstractions. However, most of these efforts focus on a single CM, which is already a hard problem. Nevertheless, there are some solutions combining together diverse CMs. Common API abstractions enable developing applications that are agnostic to the underlying CMs. Then, some local mapping is performed between the API operations and the diverse CMs/related interaction protocols supported. In this category, ReMMoC [8] is an adaptive middleware for mobile systems, enabling clients that can interact with both RPC servers and PS systems via a common programming interface. Such systems are described with extended WSDL descriptions. Same as ReMMoC, our solution also offers a common API abstraction, which however additionally covers TS systems. Following a similar approach, an API (and partial protocol stack) conforming to one CM can be locally mapped to an interaction protocol conforming to another CM. Thus in [9], the authors implement the LIME TS middleware on top of a PS substrate. Similarly, work in [10] enables Web services SOAP-based interactions over a TS binding. Contrary to these specific solutions, our approach aims to cover a much wider range of CM interoperability. Wrapping systems behind standard technology interfaces enables accessing these systems by using CMs that are different from their native ones. In [7], a gateway allows high-level access to the data and operations of a wireless sensor network via Web service interfaces. Again, our solution is much more general. Additionally, the TS model with its looser coupling offers greater flexibility in abstracting heterogeneous CMs than a service interface. Bridging is

about interworking between heterogeneous interaction protocol stacks. The Enterprise Service Bus (ESB) paradigm is currently the dominant bridging solution for the integration of heterogeneous systems, with realizations that are established industrial (open- and closed-source) products such as Apache ServiceMix and IBM Websphere ESB<sup>6</sup>. ESBs typically rely on transforming heterogeneous interaction protocols of systems plugged on the bus to a single messaging bus protocol that interconnects the endpoints representing these systems on the bus. Such transformations are carried out by adapters, which may also map between different CMs. For instance in [6], an external TS is connected through adapters to a distributed ESB topology and is accessible via the bus messaging-based interface. However, such adapters are proprietary and ad hoc, and concern each time a specific middleware platform. Recent realizations of ESBs enable multiple CMs on the bus, such as messaging and PS, which, nevertheless, are supported in parallel; hence, plugging systems with diverse coordination semantics on the bus still requires adapting to the (now richer) bus semantics. In contrast, our solution proposes a generic and systematic way for adapting between heterogeneous CMs. Acknowledging the flexibility of the TS model, a number of system integration efforts have adopted TS as the common coordination facility. Some of these approaches enrich TS with PS semantics, or offer a REST (REpresentational State Transfer)-based API in addition to the TS-based API [14]. Similar efforts introduce extended TS as an alternative solution to the realization of the ESB paradigm [15]. Some of these ESBs offer various coordination semantics (by emulating different CMs) and related APIs, such as CS- and PS- in addition to TS-based. The difference of our solution lies in using the GDS TS model as a design-time artefact, while runtime interactions are performed by employing the native CMs of the constituent systems.

## 6 Conclusion

Integrating heterogeneous systems in an orchestration workflow while preserving their native CMs is challenging. We have identified three levels of coordination (i.e., application, middleware and platform levels), for which we have incorporated generic support into a core coordination middleware. Appropriate generic programming abstractions in terms of data-driven primitives and interfaces are provided to application designers. Using these primitives, they can easily develop application workflows, focusing on only the application logic as opposed to puzzling with underlying middleware platforms of constituent systems. Middleware designers can also easily incorporate new middleware platforms on demand into the core coordination middleware. Our BPEL-based implementation of the core middleware and associated evaluation results show that our solution considerably facilitates the tasks of both designers. Certainly, there are outstanding issues regarding our solution that we are considering in our current and future work. We identified the TS model as the most appropriate one for abstracting any other CM, and we introduced GDS TS as application-level CM to be employed

<sup>6</sup> <http://servicemix.apache.org>, <http://www-01.ibm.com/software/integration/wsesb/>

in workflows. We intend to carry out a precise evaluation of the preservation (or loss) of semantics introduced by this abstraction and the related mappings between GDS and CS, PS and TS. In the same direction, we wish to evaluate the applicability of our solution in real-life use cases, which raises the issue of abstracting complex middleware platforms having rich features. We want to see how much of these features is preserved by our proposed CS, PS, TS, and further GDS abstractions, and what the related trade-off is between generic programming interfaces offering simplicity and the resulting loss of platform-specific features. On the other hand, although our model-based solution involves mostly design-time mappings and should thus introduce only small runtime overhead, we intend to accurately evaluate the performance of our coordination middleware. Finally, in the medium term, we aim to develop user-friendly tools for the application and middleware designers and investigate data-driven choreography of heterogeneous systems.

## References

1. Monson-Haefel, R., Chappell, D.: *Java Message Service*. O'Reilly & Associates, Inc., Sebastopol (2000)
2. Carzaniga, A., Wolf, A.: *Content-based Networking: A New Communication Infrastructure*. LNCS, pp. 59–68 (2002)
3. Freeman, E., Arnold, K., Hupfer, S.: *JavaSpaces Principles, Patterns, and Practice*. Addison-Wesley Longman Ltd, Essex (1999)
4. Murphy, A.L., Picco, G.P., Roman, G.C.: LIME: A Coordination Model and Middleware Supporting Mobility of Hosts and Agents. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 15(3), 328 (2006)
5. Bakshi, A., Pathak, A., Prasanna, V.: System-level Support for Macroprogramming of Networked Sensing Applications. In: *Int. Conf. on Pervasive Systems and Computing (PSC)*. Citeseer (2005)
6. Baude, F., Filali, I., Huet, F., Legrand, V., Mathias, E., Merle, P., Ruz, C., Krummenacher, R., Simperl, E., Hammerling, C., Lorre, J.P.: *ESB Federation for Large-scale SOA*. In: *Proceedings of the 2010 ACM Symposium on Applied Computing, SAC 2010*, pp. 2459–2466. ACM, New York (2010)
7. Avilés-López, E., García-Macías, J.: TinySOA: a Service-oriented Architecture for Wireless Sensor Networks. *Service Oriented Computing and Applications* 3(2), 99–108 (2009)
8. Grace, P., Blair, G.S., Samuel, S.: A reflective framework for discovery and interaction in heterogeneous mobile environments. *SIGMOBILE Mob. Comput. Commun. Rev.* 9(1), 2–14 (2005)
9. Ceriotti, M., Murphy, A.L., Picco, G.P.: Data sharing vs. message passing: Synergy or incompatibility?: An implementation-driven case study. In: *SAC 2008: Proceedings of the 2008 ACM Symposium on Applied Computing*, pp. 100–107. ACM, New York (2008)
10. Wutke, D., Martin, D., Leymann, F.: Facilitating complex web service interactions through a tuplespace binding. In: Meier, R., Terzis, S. (eds.) *DAIS 2008*. LNCS, vol. 5053, pp. 275–280. Springer, Heidelberg (2008)

11. Pietzuch, P., Eysers, D., Kounev, S., Shand, B.: Towards a common api for publish/subscribe. In: DEBS 2007: Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems, pp. 152–157. ACM, New York (2007)
12. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. *ACM Comput. Surv.* 35(2), 114–131 (2003)
13. Kay, M.: XSLT 2.0 Programmer’s Reference. Wiley Pub., Chichester (2004)
14. Nixon, L.j.b., Simperl, E., Krummenacher, R., Martin-Recuerda, F.: Tuple-space-based computing for the semantic web: A survey of the state-of-the-art. *Knowl. Eng. Rev.* 23(2), 181–212 (2008)
15. Mordinyi, R., Kühn, E., Schatten, A.: Space-based architectures as abstraction layer for distributed business applications. In: Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2010, pp. 47–53. IEEE Computer Society, Washington, DC, USA (2010)

# Wireless Sensor Networks Based on Publish/Subscribe Messaging Paradigms

Hakan Cam<sup>1</sup>, Ozgur Koray Sahingoz<sup>1</sup>, and Ahmet Coskun Sonmez<sup>2</sup>

<sup>1</sup> Turkish Air Force Academy, Computer Engineering Department,  
34149 Istanbul, Turkey

<sup>2</sup> Yildiz Technical University, Computer Engineering Department,  
34349 Istanbul, Turkey

{h.cam, o.sahingoz}@hho.edu.tr,  
acsonmez@ce.yildiz.edu.tr

**Abstract.** Rapidly increasing development and application areas of wireless sensor networks requires the deployment of highly scalable and dynamic communications paradigms for coping with the complex data management tasks encountered in distributed environments. In this context, the publish/subscribe communication mechanism plays an essential role since they provide quick and easy adaptation of the system to the dynamic nature of wireless sensor network environment. To shed a light on the design and implementation issues of publish/subscribe messaging systems we investigated how to apply publish/subscribe approach into wireless sensor networks and developed a novel application working on the network of both the real and virtual Sun SPOT sensor devices in a multi-hop manner. We present and compare the simulation results.

**Keywords:** Wireless Sensor Networks, Publish/Subscribe Messaging Systems.

## 1 Introduction

Wireless Sensor Networks (WSN) have gained increasing attention from academia and industry points of view with the helping hands of recent advances in sensor devices, processor and memory circuits, battery technologies and communication techniques in the past few years [1]. They provide reasonable and practical solutions to the complex problems of military, industrial, environment and health applications. Main problem of these applications is how to manage the transferring of raw data collected by the tiny sensor devices to the host applications over the wireless communication channels. A typical WSN comprises of hundreds or thousands of cheap and tiny sensor devices equipped with disposable battery, limited amount of wireless communication ability, processing and storage capabilities [2]. These sensor devices are scattered around the target region and collect raw data of physical phenomenon from their environments such as light, temperature, acceleration, humidity, vibration, pressure, motion, seismic value, image, noise, mechanical tension, speed, direction, etc. and transmit these collected data to the sink nodes or base stations for further processing and detailed evaluation.

In WSN architecture sensor devices capture the physical events in the environment to be monitored. Perceived events are handled separately by each sensor node. These processed events are transferred to the neighboring sensor nodes in the network coverage area via wireless communication environment. Spread over the network these processed events will be transferred to the base stations which are serving as data collection centers. These collected data will be transferred to the task management unit via internet or satellite for further analyzing and evaluation [3].

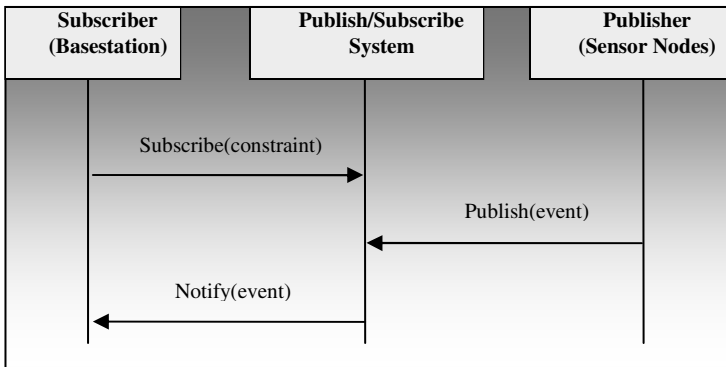
Since wireless communication radio channels of WSNs are susceptible to signal fading and interference disturbances, they have higher failure rates than wired and traditional wireless communication radio channels. Their transmission capacity is lower when compared with traditional networks, such as, based on the IEEE 802.15.4 standard, maximum bandwidth of WSNs is of 250 kb/s in the 2.4 GHz ISM band. Packet size of WSNs is shorter than traditional networks in order to be enduring to communication failures. For example maximum packet length of IEEE 802.15.4 standard is 128 bytes, whereas half of these 128 bytes could be taken away by the overhead information [4]. WSNs have a very dynamic and temporal nature of communication environment. Sensor devices are very prone to failure. Their battery power may be depleted in a short period of time, network topology may be changed at any time, communication channels are quite likely to fail and sensor devices may get out of communication range [5]. There is no need to know the address or identity of the devices that deliver the information in most of the wireless sensor network applications. This brings some relief to the problem of managing and maintaining of the addresses of a large number of sensor nodes, which is usually problematic and burdensome. Applications are more interested in the content of the data instead of sender's id or physical network address. In addition, some of the applications may have an interest in the same sensor data but with different constraints. These situations require multiple applications to work in parallel in sensor devices which goes beyond the constrained capabilities of sensor devices. It is exactly at this point where publish/subscribe messaging system [6] comes into consideration which is a variant of a data-centric communication paradigm [7] in wireless sensor networks. In this study we have investigated the integration of wireless sensor networks with a publish/subscribe system. In addition we have developed a novel application working on the network of both the real and virtual Sun SPOT sensor devices [8] in a multi-hop communication manner. The paper concludes with the presentation and comparison of simulated performance results.

The rest of the paper is organized as follows. In the next section, Section 2, we first present related works on publish/subscribe messaging systems, the rationale behind these approaches, and the advantages of using these paradigms in wireless sensor networks. Then we introduce our system model and explain how it works in Section 3. After that we present the implementation and simulation results in Section 4, and finally we conclude this paper in Section 5.



## 2 The System Architecture of Publish/Subscribe Communication Models

Publish/subscribe communication model is an important alternative paradigm for asynchronous communications between nodes in distributed systems such as wireless sensor networks [6]. There are two main components in this approach, namely the subscriber (consumer) sensor nodes and the publisher (producer) sensor nodes. In addition to these two components there are also normal sensor nodes which act like a relay. They catch a message packet and send it again without any process. The rationale behind the publish/subscribe system is that subscriber sensor nodes specify their interests in certain predicates with subscription messages and send their data packets to the neighboring sensor nodes without any knowledge and interests of producers. These subscribers will be notified after the occurrence of any event fired by the publisher sensor nodes that match their registered interests and receive these produced events [9]. Similarly, producer sensor nodes of events send their data packets to neighboring sensor nodes without knowledge and interest of consumers as shown in Fig. 1.



**Fig. 1.** Publish/Subscribe mechanism

There are four main features of publish/subscribe messaging systems. The first one is that communication is anonymous which means that both subscribers and publishers are anonymous to each other. They are not expected to know the identity of the other nodes. Therefore there are no explicit source and destination addresses. The second one is that communication is inherently asynchronous which means that there is no need of communication between the two sensor nodes to occur simultaneously. Since there is no acknowledgement mechanism between source and destination, the packet sender doesn't have to wait for a reply or an acknowledgement from the packet receiver. It proceeds on to its other duties after it sends the packets so neither packet senders nor packet receivers are blocked. The third one is that it allows multicast or broadcast communication which means that when the packet sender transmits the packet it sends the same packet to many receivers within the transmission range at the same time. The last feature is that though sensor nodes may connect and disconnect to

and from the network at any time, publish/subscribe system can have adaptation capability to handle connectivity problems of dynamically changing environment [10]. Despite there are many advantages of publish/subscribe messaging paradigm, there are also some shortcomings. Some of the key technical challenges in implementing publish/subscribe messaging paradigm over WSNs are coarsely listed below: One of the difficulty is that when a sensor node fails it is not possible to send keep-alive messages used for detecting failures of sensor nodes and communication radio channels. Second one is that to manage successfully duty cycle of sensor nodes. Sensor nodes sleep when they are inactive to preserve their constrained power. When they have sensed new data they became active and publish this new version of data. The longer the inactive time, the more difficult to be informed of publish/subscribe system to communicate with inactive sensor nodes.

In the past few years there has been much research on the publish/subscribe systems in the wired and wireless networks [11, 12], but not so much on the wireless sensor networks. Publish/subscribe messaging systems provide many advantages to wireless sensor network applications, which were described in the previous paragraph. As explained previously this messaging approach inherently meets many requirements for data communication in wireless sensor networks such as hiding the complex topology of the network from the application programmers/designers and allowing data packets to be delivered based on sensor nodes' interests rather than their addresses. Therefore publish/subscribe systems provide ideal solutions for deployment of highly scalable and dynamic communication paradigms like wireless sensor networks.

Content-based, topic-based and type-based systems are the three main types of publish/subscribe messaging systems [6]. Among all of the three messaging systems the content-based messaging systems are the most powerful, best known and widely used models. In this structure the subscriber expresses its interest predicate or filtering criteria with the content of messages it wants to receive. For instance the content of this message can be a certain threshold value of battery voltage, battery discharge rate, 3D acceleration, temperature or light sensed from the environment. In the topic-based systems subscriber events or publisher events can only be made on a specified set of topics which are usually predefined in a list. Subscriptions contain only the name of a class of messages chosen among a set of predefined classes. In the type-based messaging systems, a subscriber expresses the type of data it is interested in, such as acceleration, temperature and light value sensed from the environment.

Some of the research carried out about the publish/subscribe systems in the wireless sensor networks is presented briefly in the following section.

In [13], the design and implementation issues of publish/subscribe messaging paradigm are presented. With this component framework application programmers can have the ability of adapting the model by making orthogonal choices about the communication protocol components for subscription and notification delivery, the supported data attributes and a set of service extension components. Attribute-based naming scheme is used in this framework and different addressing schemes and interaction patterns are supported.

In [14], various implementations of publish/subscribe messaging paradigms are investigated in terms of basic network primitives and data centric storage mechanisms. Performance and functionality of all the possible combinations of publish/subscribe

paradigm and communication primitives are considered. The main concern is how various publish/subscribe paradigms can be implemented over basic unicast or broadcast communication mechanisms.

In this section of the paper we compare some of the basic primitives of our system with the two previously mentioned systems. In [13], the authors claim that their component framework is the variant of TinyCOPS framework and is implemented on eyesIFX and Tmote Sky sensor nodes. Their framework is carried out in TinyOS 2.0 operating system. They used TWIST multi-platform testbed for obtaining and testing the experimental results. In [14], the authors chose ns-2 discrete event simulator environment for evaluating the performance of their framework using OTcl Scripts and C++ language. They have just studied basic unicast or broadcast mechanisms, but multicast mechanisms were not included in their framework. Unlike these features, our model is implemented on both real and virtual Sun SPOT wireless sensor nodes. Our framework works on the J2ME Platform for Squawk Java Virtual Machine. Our application is developed with Net Beans 6.8 IDE and Java programming language. We chose Spot World Solarium Emulator testbed for our virtual Sun SPOT wireless sensor nodes to evaluate the performance of our system.

### 3 System Model

The proposed system architecture is composed of two different applications. The first application is subscriber application and works on the host side of the network. The second application is publisher application and works on the real/virtual Sun SPOT wireless sensor nodes. The host application communicates with the sensor nodes via base station which is connected with a USB cable to the computer. Subscriber application and publisher application are depicted in Fig. 2, and Fig. 3, respectively.

In this model both subscriber application and publisher application establish a radiogram connection by using radiogram protocol. Radiogram protocol is a datagram-based protocol that allows the exchange of packets between two sensor devices. A data

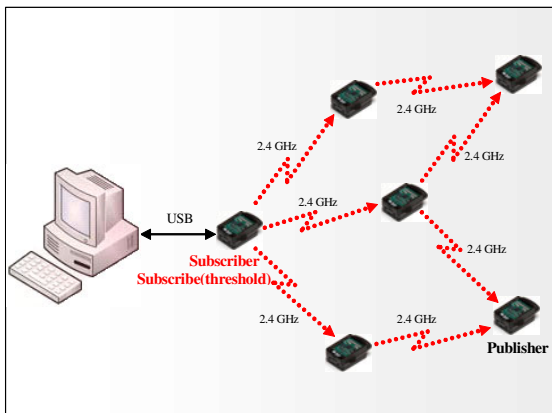


Fig. 2. Subscribe mechanism

packet is sent between the two sensor devices in datagrams, of type Datagram. This protocol provides no guarantees about packet delivery or packet ordering. Datagrams are sent over more than one hop could be silently lost, be delivered more than once, and could be delivered out of sequence. This protocol is implemented on top of the 802.15.4 MAC layer [15].

Subscriber application resides on the host application and transmits the threshold packet to the publisher sensor nodes via base station over the wireless sensor network. This threshold packet contains the values of interest of the subscriber application. After sending this subscription packet it starts to listen if the publisher data packages will come to it containing the threshold values of interest. When it receives a packet from a publisher node it compares the threshold value with the incoming data value and if the condition is met it displays the details of the publisher sensor node packet, if not it just discards it.

These subscription threshold packets are transmitted by the base station to the sensor nodes within the transmission range. When a sensor node receives this threshold packet it fulfills four different tasks. First, it retransmits this packet to all of its neighbors within its transmission range. Second, it compares this threshold value with its sensor data values table created with sensor values sensed from the environment. Third, if the condition is met it generates a publisher sensor data packet which can contain sensing battery voltage, battery discharge rate, light, temperature, x, y, z acceleration values, IEEE extended MAC address and transmits this packet to all of its neighbors within its transmission range. Finally, it creates its routing table according to incoming subscription messages. If a sensor node receives a previously transmitted data packet it discards it to avoid transmitting unnecessary data packets and entering infinite loops. This process goes on until the subscriber threshold packet reaches to every sensor nodes over the network.

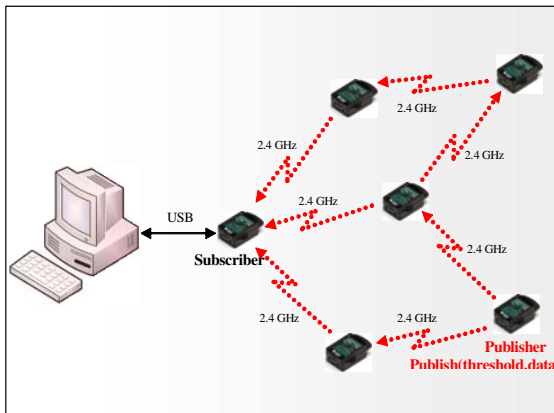


Fig. 3. Publish mechanism

## 4 Performance Analysis

A novel publish/subscribe messaging system has been developed above, which is an efficient communication method to send data packets between sensor nodes over

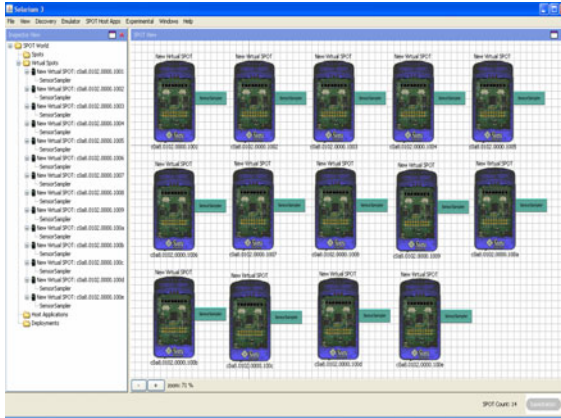
wireless sensor networks. This system developed with Sun SPOT wireless sensor network kit [8], was implemented and tested on both real/virtual sensor devices. We have chosen Sun SPOT sensor devices because we believe that these devices are ideal for integrating publish/subscribe messaging systems into wireless sensor networks. Sun SPOT sensor device is built by stacking a Sun SPOT processor board with a sensor board and a battery. It works on the Squawk Java Virtual Machine (JVM) [16]. Some of the features of Sun SPOT sensor devices are listed in the Table 1.

We developed our application using Net Beans 6.8 IDE with Java programming language by using J2ME Platform for a Squawk JVM. Spot World Solarium Emulator [8] is used for virtual Sun SPOT sensor devices to get the test results of this system. We placed 14 Virtual Sun SPOT sensor devices in the Emulator as shown in Fig. 4. We compiled, built and deployed our application to the virtual sensor devices and ran it. The range of transmission coverage among virtual sensor devices was chosen as 10 m. The data communication between sensor nodes was designed to take place in a multi-hop manner.

In the following section we evaluate and present the cost of the publish/subscribe operations which can be expressed in terms of the number of IEEE 802.15.4. MAC layer data packets that are transmitted, forwarded or received by each sensor device. We consider that these operations are the most important causes of energy consumption by the wireless radio in the wireless sensor networks. In the first implementation scheme (scheme\_1) when a sensor node receives a data packet it transmits the data packet to the neighboring nodes within the communication range.

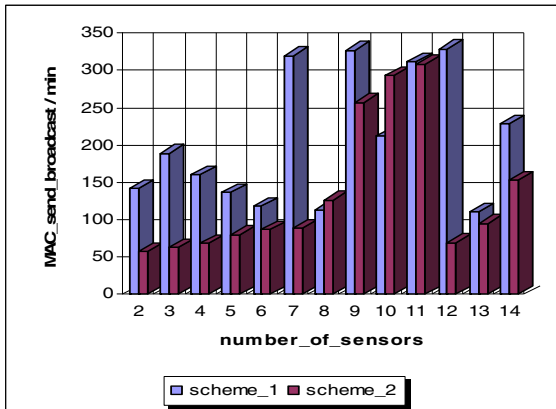
**Table 1.** Features of Sun SPOT sensor devices

<b>Software</b>	<b>Squawk Virtual Machine</b>	<ul style="list-style-type: none"> <li>* Fully capable J2ME CLDC 1.1 Java VM with OS functionality</li> <li>* VM executes directly out of flash memory</li> <li>* Device drivers written in Java</li> <li>* Automatic battery management</li> </ul>
	<b>Battery</b>	<ul style="list-style-type: none"> <li>* 3.7V rechargeable 720 mAh lithium-ion battery</li> <li>* 32 uA deep sleep mode</li> </ul>
	<b>Sensor Board</b>	<ul style="list-style-type: none"> <li>* 2G/6G 3-axis Accelerometer</li> <li>* Temperature sensor, Light sensor</li> <li>* 8 tri-color LEDs</li> <li>* 6 analog inputs</li> <li>* 2 momentary switches,</li> <li>* 5 general purpose I/O pins</li> </ul>
<b>Hardware</b>	<b>Processor Board</b>	<ul style="list-style-type: none"> <li>* 180 MHz, 32 bit ARM920T core</li> <li>* 512K RAM, 4M Flash memory</li> <li>* 2.4 GHz IEEE 802.15.4 radio with integrated antenna</li> <li>* USB interface</li> </ul>



**Fig. 4.** Deployment of virtual Sun SPOT sensor devices in the Solarium Emulator

In the second implementation scheme (scheme\_2) sensor nodes transmit data packets only once to avoid packet collisions and to facilitate effective energy consumption through decreasing the packet traffic. The total number of packets transmitted by every sensor nodes in these two schemes in one minute period is depicted in Fig. 5. According to this figure, in scheme\_1 the number of packets transmitted by each sensor node grows for a while, but with the increasing number of sensors, receiving sensor nodes are overwhelmed with data packets from transmitting nodes. This produces packet collisions and degrades network performance. This case can be seen as a fluctuation in the number of packets in the figure. When a packet collision occurs the packets are discarded due to the heavy traffic load and limited processing power. Since each sensor node transmits the packets only once, less packet transmissions occur in scheme\_2 compared with scheme\_1. In this case receiving sensor nodes are not overwhelmed with data packets from transmitting nodes. This results in less energy consumption of sensor nodes and extends the network lifetime.



**Fig. 5.** Total number of transmitted signals

The total number of packets received by each sensor node in these two schemes in one minute period is depicted in Fig. 6. In this figure, it can be seen that the number of packets received by each sensor node in scheme\_1 fluctuates compared with scheme\_2. In scheme\_1 each sensor node transmits and receives every packet in the network and due to the heavy packet traffic load, packet collisions occur, too. In scheme\_2, each sensor node receives every packet in the network only once, so less packet traffic load and less packet collisions occur. Therefore energy consumption of sensor nodes decreases and the network lifetime increases.

When sensor nodes with constrained processing power transmit and receive more data packets, more packet traffic load and more packet collisions take place. This leads to consuming more energy and depleting the battery power of sensor nodes. Accordingly, scheme\_2 provides better solution for energy consumption and longer network life-span than scheme\_1.

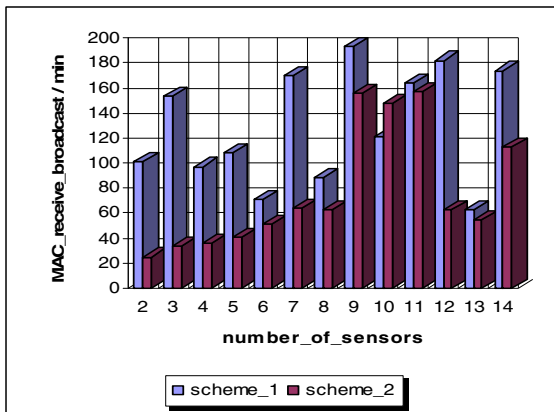


Fig. 6. Total number of received signals

## 5 Conclusion

In this paper we discussed how to apply publish/subscribe messaging systems into wireless sensor networks and developed a novel application working on the network of both the real and virtual Sun SPOT sensor devices in a multi-hop manner. We described the architecture and the rationale behind the publish/subscribe systems for shedding light on the design and implementation issues of publish/subscribe messaging systems. The publish/subscribe model has already been playing an essential role and has promising solutions in the future of wireless sensor networks. They provide better solutions for energy consumption and network lifetime in wireless sensor networks. Simulation results of our proposed system model have demonstrated the expected results.

Our ongoing research efforts are being devoted to the integration of wireless sensor networks with multi-agent systems [17] for solving some of the energy, security, routing and data aggregation problems effectively encountered in dynamic and distributed environments like wireless sensor networks.

## References

1. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. *Computer Networks* 52(12), 2292–2330 (2008)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. *IEEE Communications Magazine* 40(8), 104–112 (2002)
3. Baronti, P., Pillai, P., Chook, V., Chessa, S., Gotta, A., Fun Hu, Y.: Wireless Sensor Networks: a survey on the state of the art and the 802. *Computer Communication* 30(7), 1655–1695 (2007)
4. Hunkeler, U., Truong, H.L., Clark, A.S.: MQTT-S-A Publish/Subscribe Protocol For Wireless Sensor Networks. In: 2nd Workshop on Information Assurance For Middleware Communications, IAMCOM 2008, India (January 2008)
5. Chong, C.Y., Kumar, S.P.: Sensor Networks: Evolution, Opportunities, and Challenges. *Proceedings of the IEEE* 91(8), 1247–1256 (2003)
6. Eugster, P.T., Felber, P.A., Guerraoui, R., Kernmarrec, A.M.: The many faces of publish/subscribe. *ACM Computing Surveys* 35(2), 114–131 (2003)
7. Estrin, D., Govindan, R., Heidemann, J.S., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: *Mobile Computing and Networking*, pp. 263–270 (1999)
8. Sun<sup>TM</sup> Small Programmable Object Technology (Sun SPOT), <http://www.sunspotworld.com/>
9. Liu, N., Liu, M., Zhu, J., Gong, H.: A Community-Based Event Delivery Protocol in Publish/Subscribe Systems for Delay Tolerant Sensor Networks. *Sensors* 9, 7580–7594 (2009)
10. Huang, Y., Garcia-Molina, H.: Publish/Subscribe in a Mobile Environment. *Wireless Networks* 10(6), 643–652 (2004)
11. Zheng, Y., Cao, J.N., Liu, M., Wang, J.L.: Efficient Event Delivery in Publish/Subscribe Systems for Wireless Mesh Networks. In: *Proceedings of Wireless Communications and Networking Conference, Hong Kong, China* (2007)
12. Bholra, S., Strom, R., Bagchi, S., Zhao, Y., Auerbach, J.: Exactly-Once Delivery in a Content-Based Publish-subscribe System. In: *Proceedings of Dependable Systems and Networks, USA* (2002)
13. Hauer, J.-H., Handziski, V., Köpke, A., Willig, A., Wolisz, A.: A Component Framework for Content-Based Publish/Subscribe in Sensor Networks. In: Verdone, R. (ed.) *EWSN 2008. LNCS, vol. 4913*, pp. 369–385. Springer, Heidelberg (2008)
14. Albano, M., Chessa, S.: Publish/subscribe in wireless sensor networks based on data centric storage. In: *4th International Conference on Communication System Software and Middleware (COMSWARE), Dublin, Ireland. ACM International Conference Proceeding Series, vol. 385*, pp. 37–42 (2009)
15. IEEE LAN/MAN Standards Committee, IEEE Standard 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY), Specifications for Low-Rate Wireless Personal Area Networks (2003), <http://www.ieee802.org/15/pub/TG4.html>
16. Simon, D., Cifuentes, C.: The Squawk Java Virtual Machine: Java on the Bare Metal. In: *Proc. of the 20th Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2005), USA, October 16-20* (2005)
17. Cam, H., Sahingoz, O.K., Sonmez, A.C.: The Integration of Wireless Sensor Networks with Multi-agent Systems. In: *1st Avionics and System Integration Symposium, ASES 2010, Turkish Air Force Academy, Istanbul, Turkey, April 29-30* (2010) (in Turkish)



# Application-Centric Connectivity Restoration Algorithm for Wireless Sensor and Actor Networks

Muhammad Imran<sup>1</sup>, Abas Md. Said<sup>1</sup>, Mohamed Younis<sup>2</sup>, and Halabi Hasbullah<sup>1</sup>

<sup>1</sup>Dept. of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia

<sup>2</sup>Dept. of Computer Science & Electrical Eng.,

University of Maryland Baltimore County, USA

cmimran81@yahoo.com, abass@petronas.com.my, younis@umbc.edu

**Abstract.** This paper presents ACR, a novel hybrid application-centric connectivity restoration algorithm that factors in application level interests besides efficient resource utilization while recovering from critical node failures. As a pre-failure planning measure to minimize recovery delay, ACR identifies primary actors that are critical for network connectivity based on localized information and designates them for backup nodes. The backup nodes are carefully picked to satisfy application level concerns such as high actor effectiveness. In order to minimize the impact of critical node failure on coverage and connectivity, ACR appoint high degree nodes with overlapped coverage. Upon failure detection, the pre-designated backup pursues controlled and coordinated movement to replace the failed node. Simulation results validate the performance of ACR.

**Keywords:** Wireless sensor and actor network, Fault tolerance, Connectivity restoration, Controlled and coordinated mobility.

## 1 Introduction

Wireless Sensor and Actor Networks (WSANs) [1] are gaining growing interest because of their suitability for mission critical applications that require autonomous and intelligent interaction with the environment. Examples of these applications include forest fire detection and containment, disaster management, search and rescue, battlefield reconnaissance etc. In these critical WSAN applications, actors establish and maintain inter-actor topology in order to collaborate with each other to plan an optimal coordinated response, synchronize their operations and respond to events. Each actor is equipped with limited resources and capabilities for executing a critical task. Failure of a critical actor may split the inter-actor network into disjoint segments while leaving some regions uncovered. Consequently, an inter-actor interaction may cease and the network becomes incapable of delivering a timely response to a serious event that causes major failures at application.

In this paper, we present a novel hybrid Application-centric Connectivity Restoration (ACR) algorithm that factors in application level concerns besides minimizing recovery time and overhead while repairing damaged topology. ACR determines critical actors primary and designates for them backup nodes as part of pre-failure planning.

Each critical actor (primary) picks a suitable backup that can satisfy application level constraints. While choosing a backup, a primary actor strives to find a nearby non-critical backup node in order to limit the scope of recovery and reduce the overhead. Moreover, ACR strives to minimize the affect of actor failure on coverage and connectivity by engaging strongly connected nodes with overlapped coverage. The pre-assigned backup pursues controlled and coordinated motion to reach the position of a failed primary. Since moving a critical backup actor may further break the inter-connectivity, ACR is recursively applied until all actors become connected. To the best of our knowledge, this is the first hybrid algorithm that considers application-level interests while reducing the recovery overhead, in addition to reducing the impact of critical actor failure on coverage and connectivity. The simulation results confirm the effectiveness of ACR in terms of satisfying application concerns and limiting the incurred overhead

This paper is organized as follows. Section 2 discusses the system model and problem statement. Related work is discussed in Section 3. The proposed ACR algorithm is detailed in Section 4. The performance of ACR is evaluated in Section 5. Finally, Section 6 concludes the paper.

## 2 System Model and Problem Statement

An actor is assumed to be able to move on demand and before moving it informs its backup so that it may not be wrongly perceived as faulty. An actor is aware of the positions of its 1-hop neighbors, e.g., by applying GPS-free localization schemes [6]. An actor failure may cause degraded task execution, drop in coverage and severed connectivity. Actor capabilities and its current task may determine the significance of an actor from application and coverage perspective. Similarly, the position of an actor significantly affects the inter-actor connectivity. For example, losing a leaf/non-critical node, such as  $A_5$  in Figure 1, does not affect inter-actor connectivity. Meanwhile, the failure of a critical actor such as  $A_2$  partitions the network into disjoint segments. ACR pursues actor relocation to recover from critical node failures. We consider one failure at a time and assume that no node fails during the recovery of another.

We associate two application-level parameters to each actor, i.e., Actor Capabilities (AC) and Task Criticality Index (TCI). Each actor would maintain the value of AC and TCI in the range [0-1]. The value of AC determines the application aspect, i.e., what an actor is expected to do. The lower bound 0 is interpreted as actor's inability to respond, whereas, 1 means actor can fully respond to an event in the area covered by an actor. Moreover, TCI refers to the priority of the current task being executed by the actor where 1 means actor is executing an extremely important task. A noticeable point is that AC has a higher priority than TCI since it reflects application-level, multi-task-based, aspect. In addition to these two values, actors periodically exchange ID, location and degree with their 1-hop neighbors.

## 3 Related Work

The existing mobility control approaches to mitigate the impact of critical node failure can be categorized into: (i) proactive (ii) reactive and (iii) hybrid. Proactive

approaches [7-8] provision fault tolerance by employing redundant nodes to establish and maintain bi-connected topology. Proactive approaches necessitate large actor count that leads to higher cost and becomes impractical. On the other hand, reactive approaches [2-5] orchestrate recovery once the failure has occurred. Reactive approaches might not be suitable for time-sensitive applications. Like ACR, PCR [11] pursues hybrid approach in order to restore connectivity. However, PCR does not consider application-level constraints on mobility of actors. Generally, hybrid recovery schemes better suit autonomous WSANs that are deployed for time-sensitive applications.

The existing node recovery techniques pursued in different contexts can be classified based on the objective function. Most of the existing schemes either consider coverage [10] or connectivity [2, 3]. A number of schemes care for both connectivity and coverage. For example, C<sup>3</sup>R [4], VCR [5], PCR [9], etc. employ node relocation to cope with the loss of coverage and connectivity when an actor fails. Unlike [2-5, 9-10], ACR factors in application-level constraints on the mobility of actors besides coverage and connectivity. The closest work to ACR is C<sup>2</sup>AM, proposed in [11], where C<sup>2</sup>AM factors in application constraints on the mobility of actors. However, there are a few differences. First, C<sup>2</sup>AM is purely a reactive approach for connectivity restoration. Second, it does not consider actor capabilities in the recovery process. Third, C<sup>2</sup>AM does not care for actor coverage.

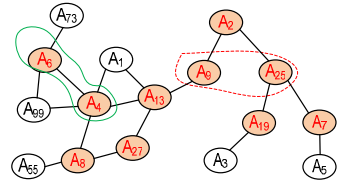
## 4 Application-Centric Connectivity Restoration Algorithm

As stated earlier, hybrid algorithms better suit resource-constrained time-sensitive applications. Therefore, unlike contemporary schemes found in the literature, the proposed ACR algorithm plans recovery ahead of time (i.e., proactive) and executes it in response to a failure (i.e., reactive). The main idea is to identify critical nodes primary in the network and appoint appropriate backup for them, preferably among the non-critical nodes during the network bootstrapping phase. Once the failure of the primary is detected through missing heartbeats, the backup immediately initiates a recovery that involves reconfiguring the topology. The detailed algorithm is described in following subsections.

### 4.1 Determining Cut-Vertex (Critical) Actors

As described earlier, the failure of critical actor divides the inter-actor network into disjoint segments in addition to leaving a coverage hole. Therefore, ACR determines critical nodes as part of pre-failure planning and designates for them backup actors to tolerate node failures. ACR employs a simple localized cut-vertex detection procedure that only requires 1-hop positional information to detect critical nodes. The procedure is based on [12] and runs on each node in a distributed manner to determine locally whether a node is critical or not. An actor is 1-hop critical if its 1-hop neighbors can be partitioned into more than one segment, non-critical otherwise.

Figure 1 shows the critical (shaded circles) and non-critical nodes. For instance, Figure 1 also shows a localized view of critical actor  $A_2$  (dotted line) and non-critical node  $A_{99}$  (solid line). Node  $A_2$  is 1-hop positional critical since its 1-hop neighbors  $A_9$  and  $A_{25}$  become disconnected without  $A_2$ . whereas, neighbors of  $A_{99}$ , i.e.,  $A_4$  and  $A_6$  remain connected, therefore, it is 1-hop positional non-critical node. Moreover, leaf nodes such as  $A_3$ ,  $A_5$ , etc. are detected as non-critical, since there, failure does not inflict inter-actor connectivity.



**Fig. 1.** A connected inter-actor network with critical and non-critical actors

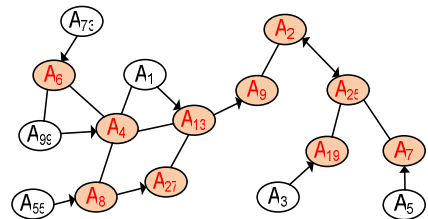
### 4.1 Backup Selection and Failure Detection

Once a critical actor is identified, it chooses an appropriate backup to handle its failure. Each primary preferably picks a non-critical healthy backup among 1-hop neighbors based on its impact on application, coverage and connectivity.

Selection of a backup: The actors maintain minimum state information (i.e., 1-hop neighbors) to avoid excessive messaging overhead, since with 1-hop information, neighbors of the failed critical actor become disconnected and cannot coordinate. Therefore, backup actors are determined and notified before a failure of critical nodes takes place. Consider the inter-actor topology presented in Figure 2 and assume parameter values in Table 1 to have better understanding of the procedure. The selection of a backup among 1-hop neighbors is based on the following ordered criteria:

Neighbor Position (NP): As discussed above, each actor determines whether it is critical or non-critical depending on the position of that node in the topology. A non-critical neighbor actor is preferred to serve as backup because it will limit the scope of recovery that ultimately reduces the implication on application, coverage and connectivity. For example, critical actor  $A_8$  prefers to appoint non-critical node  $A_{55}$  as backup instead of critical actors  $A_4$  and  $A_{27}$  as shown in Figure 2. The arrow head points towards the primary (critical) nodes.

Application-level interests: A non-critical neighbor with most appropriate actor capabilities (AC) and/or executing non-essential task (least TCI) is more suitable candidate for backup. Choosing an unsuitable node will be a futile effort because it cannot respond to an event as expected. Moreover, moving an actor executing least TCI will have minimum implication on application-level task. Unlike PCR [9], ACR prefer to choose as a backup a non-critical node among the 1-



**Fig. 2.** Critical actors designate their backup based on the criteria specified

hop neighbors with similar AC and least TCI. As shown in Figure 2, node  $A_2$  picks  $A_{25}$  as backup because of higher AC than  $A_9$ . Similarly, actor  $A_{27}$  choose node  $A_8$  as backup due to least TCI than  $A_{13}$ .

**Connectivity:** An actor that causes minimum disturbance to application tasks while having strong connectivity is better choice to serve as backup. Choosing a strongly connected node most probably has non-critical actors in the neighborhood. Moreover, moving such a node will improve the overall connectivity of the network in addition to limiting the scope of recovery. On the other hand, moving weakly connected nodes may trigger successive cascaded relocations that significantly increase the movement overhead. In contrary to DARA [2] and PCR [9], ACR appoint higher degree nodes. For example, a cut-vertex  $A_9$  prefers to designate actor  $A_{13}$  as backup over  $A_2$  due to higher degree as shown in Figure 2.

**Overlapped coverage:** A strongly connected node has more neighbors that increase the possibility of having actors with more overlapped coverage. A high degree with more overlapped coverage is preferred to serve as backup. Moving a node with more overlapped coverage will mitigate the effect of the lost actor without major degradation of the coverage in other parts of the network.

It is to be noticed that ACR pursues localized greedy heuristic that may not always leads to optimal solution. For instance, choosing actor  $A_{13}$  as backup for node  $A_{27}$  instead of  $A_8$  would result in overall least TCI. However, it would have required more network state information that is not feasible to maintain. Nonetheless, simulation results have shown that ACR significantly outperforms DARA although it maintains more network state information as will be discussed in Section V.

**Primary monitoring and Failure Detection:** Neighbor actors exchange heartbeat messages as part of their network operation to update their status. The chosen backup actors are notified via these messages. Once an actor receives BACKUP notification, it starts monitoring the primary through heartbeats. Missing a number of consecutive heartbeats is perceived by backup as failure of primary. For instance, a backup node  $A_{25}$  detects the failure of primary  $A_2$  shown in figure 3(a) and initiates a recovery process as detailed in the following section.

#### 4.1 Failure Recovery

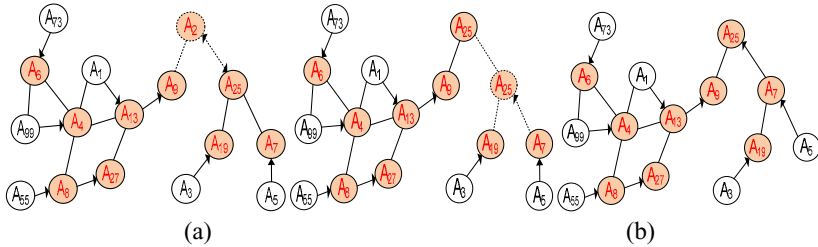
The pre-designated backup actor immediately initiates a recovery process once it detects the failure of the primary. Three scenarios may be encountered. First, if the

**Table 1.** Parameter values of actors in Figure 3

ID	NP	AC	TCI	Degree
A1	N	4	2	2
A2	C	4	2	2
A3	N	4	3	1
A4	C	4	4	5
A5	N	2	3	1
A6	C	1	2	3
A7	C	4	3	2
A8	C	4	1	3
A9	C	3	3	2
A13	C	4	2	4
A19	C	4	4	2
A25	C	4	3	3
A27	C	3	2	2
A55	N	4	4	1
A73	N	4	2	1
A99	N	4	1	2

backup actor is critical then it checks whether the failed node was also its backup or not, i.e., the two nodes are backup for each other. In Figure 2, nodes  $A_{25}$  and  $A_2$  are serving as each other backup. Now the backup actor chooses and appoints another backup using the same criteria as specified in preceding section. For example, Figure 3(a) shows that actor  $A_{25}$  designates node  $A_7$  as its new backup..  $A_{25}$  sends a movement notification message to newly appointed backup so that it can maintain its connectivity with primary. Once the backup is notified, the primary moves to the location of failed node and starts exchanging heartbeat messages with new neighbors as shown in Figure 3(b). However, moving a critical node further partitions the network, therefore, algorithm is recursively executed on notified backup until non-critical node is reached. Figure 3(b) shows that moving critical actor  $A_{25}$  further partitions the network and algorithm is recursively applied until the connectivity is restored or network periphery is reached. The backup nodes successively replace their primary in a *cascaded* manner. The recovery process is similar for both the cases whether the primary node fails or moves as part of recovery.

Second, if the pre-designated backup actor is critical and its backup is alive then it just send a movement notification message to backup and move to the location of failed or moved actor as shown in Figure 3(c). Third, if the backup is non-critical then it simply replaces the primary and recovery is complete as shown in Figure 3(c). The pseudo code of ACR is omitted due to space constraints.



**Fig. 3.** The failure detection and recovery procedure executed by ACR for the network segment shown in Figure 2; (a) the primary node  $A_2$  fails and is detected by pre-designated backup actor  $A_{25}$  (b) Backup  $A_{25}$  choose another backup (since failed node was its backup), send movement notification message to newly appointed backup and moves to location of  $A_2$ (c) The backup node  $A_7$  replace the primary  $A_{25}$ , whereas, non-critical backup  $A_5$  replaces the primary  $A_7$  to complete the recovery.

## 5 Results and Analysis

The performance of ACR is validated through extensive simulations. This section describes the simulation environment, performance metrics and experimental results.

### 5.1 Simulation Setup and Performance Metrics

The experiments involve randomly generated topologies in an area of  $1000m \times 600m$  with varying actor count and communication range. The number of actors has been

set to 20, 40, 60, 80 and 100. The communication range of actors is changed among 50, 75, 100 and 125. When changing the node count, “ $r$ ” is fixed at 100m; and “ $N$ ” is set to 100 while varying the communication range. The values of AC and TCI are randomly assigned to actors using discrete uniform distribution in the range [0, 5]. We choose a critical actor at random to be failed. The results of individual experiments are averaged over 30 trials. All results are subject to 90% confidence interval analysis and stays within 10% the sample mean. The performance of ACR is assessed using the following metrics:

- *Max change in AC*: This metric captures the variations in the AC caused by swapping of actor positions. It reports the maximum change in the AC when a node replaces another node. This includes the backup and subsequent relocation until the recovery algorithm terminates. This metric in essence indicates the readiness of the network to handle serious events in vicinity of a replaced actor given the capabilities of the node that moved in. ACR strives to move more capable actors with respect to the failed one so that the on-going network operation should be sustained effectively.
- *Average TCI*: measures the average TCI of all the nodes participating in the recovery. This metric reflects the level of disturbance caused to critical tasks.
- *Total movement distance*: reports the total distance moved by all actors during recovery: This gauges the efficiency of the ACR algorithm in terms of energy efficiency, recovery time and overhead.
- *Number of actors moved during the recovery*: This metric reflects the scope of the recovery which indicates the level of disturbance to the network operation.
- *Number of coordination messages exchanged*: Again this metric indicates the energy consumption and recovery overhead in terms of communication.
- *Percentage of Area coverage reduction* relative to the pre-failure level: assesses how effectively ACR limits the coverage loss while appointing backup actors.

The following parameters were used to vary the WSA configuration in the simulation experiments and study the impact on performance of ACR:

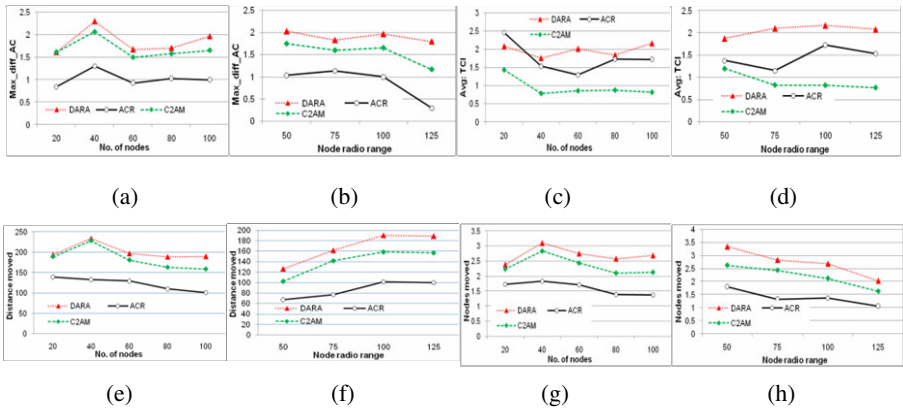
- *Number of placed actors ( $N$ )*: This parameter affects the actor density and the inter-actor connectivity. Boosting the actor density increases the number of non-critical nodes in addition to growing the area coverage.
- *Actor communication range ( $r$ )*: The communication range influences the inter-actor connectivity and highly affects the recovery overhead in terms of the traveled distance and the number of involved actors.

The performance of ACR is compared to DARA [2] and C<sup>2</sup>AM [15]. Like ACR, both the algorithms exploit actor mobility to recover from node failures. However, DARA and C<sup>2</sup>AM are reactive approaches that replace a failed node  $F$  with one its suitable neighbor and continue successive relocations until connectivity is restored or the network periphery is reached. DARA does not factor in the application-level interest at all; whereas, C<sup>2</sup>AM only consider the importance of currently-executed task. Neither DARA nor C<sup>2</sup>AM consider the actor capability and coverage.

### 5.2 Results and Analysis

**Max change in AC:** Figure 4 (a-b) confirms the effectiveness of ACR over other contemporary schemes in terms of considering application-level concerns. Basically, ACR avoids making major changes in the acting capabilities in a particular region. ACR strives to pick a backup with as close AC value to the primary node. Both graphs indicate that ACR consistently outperforms application-oblivious schemes in terms of caring for application interests while varying the actor density and communication range. This is because increasing the number of actors leads to stronger inter-actor connectivity and hence, more candidate actors would be available around the failed node. This allows ACR to designate a nearby actor with suitable capabilities. Figure 4(b) further confirms our inference.

**Average TCI:** Figure 4 (c-d) reports the average TCI for the actors involved in the recovery. The plot in essence indicates the application-level disturbance caused due to moving nodes during the recovery. Since the primary concern of C<sup>2</sup>AM is to minimize disruption to on-going operation, it performs better than the other schemes. Figure 4(c) suggests that the performance of ACR surpasses that of DARA especially for larger value of N. The obvious reason is ACR’s priority of moving non-critical nodes with appropriate actor capabilities and least TCI. Figure 4(d) confirms the effectiveness of ACR over other application-unaware schemes in terms of interrupting critical tasks while varying the transmission range. This is due to the high node density that increases the number of neighbors. Moreover, increasing ‘r’ further boosts the node degree, which enhances the prospect for picking a more suitable backup. The figure might give an impression that the performance of ACR becomes worse with the increased transmission range. In fact, this is due to ACR’s preference of limiting the recovery scope and balanced utilization of actor capability as can be observe from Figure 4 (a-b) and (g-h).



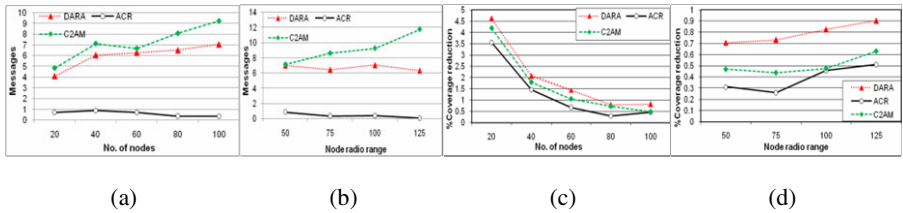
**Fig. 4.** Max change in AC as a function of  $N$  (a) and  $r$  (b). Impact on TCI as a function of  $N$  (c) and  $r$  (d). Total movement distance as a function of  $N$  (e) and  $r$  (f). Number of nodes moved as a function of  $N$  (g) and  $r$  (h).



Total movement distance: Figure 4 (e-f) shows the total distance moved by all nodes until the connectivity is restored. As the Figure 4(e) indicates ACR consistently outperforms the baseline schemes, especially for sparse networks. This is because ACR designates high-degree nodes as backup which increases the probability of having non-critical nodes in the neighborhood. Thus, it strives to avoid successive cascaded relocations. Figure 4(e) suggests that despite considering application constraints, the performance of ACR scales very well and is not affected by the node density because of choosing non-critical nodes as backup. While varying the transmission range, ACR incurs far less overhead than other schemes. Again, this is due to limiting the scope of cascaded relocations by choosing non-critical actors as shown in Figure 4(f). Moreover, the performance of ACR is not much affected by increasing the communication range despite ACR’s concern about application-level constraints in addition to minimizing number of nodes involved in recovery as will be later discussed. The performance of DARA and C<sup>2</sup>AM worsens with the growth in the transmission range because of the increased distance between nodes.

Number of nodes moved: Figure 4 (g-h) shows the number recovery participants when ACR and the baseline approaches are applied. The performance graphs confirm the advantage of ACR which moves fewer actors than all the other approaches. This is because ACR limits the scope of the recovery and avoids successive cascaded relocations by choosing non-critical nodes as backup. Moreover, ACR designates high degree nodes as backup that have more probability to have non-critical nodes in the neighborhood. Furthermore, the performance of ACR improves with the high actor density and communication range that indicates great scalability.

Number of coordination messages: Figure 5 (a-b) reports on the coordination messaging overhead as a function of the network size and radio range. As expected, ACR incurs far less messaging overhead than DARA and C<sup>2</sup>AM. This is because ACR maintains 1-hop information and strives to engage non-critical nodes as backup which do not require coordination messages. Furthermore, ACR appoints highly connected nodes as backup which increases the possibility of having non-critical nodes in the neighborhood. This limits the cascaded relocations and thus reduces the number of coordination messages. Unlike DARA and C<sup>2</sup>AM, the performance of ACR is improved with the high actor density and communication range. As both figures indicates C<sup>2</sup>AM incurs slightly more messaging overhead than DARA. At first instant, it seems surprising but the matter of fact is that C<sup>2</sup>AM strives to look for best candidates having least TCI which are often found in the very active parts of the network. Therefore, it requires more messaging as the number of neighbors increases.



**Fig. 5.** The effect of changing  $N$  (a) and  $r$  (b) on total number of coordination messages

exchanged. The coverage reduction after recovery, as a function of  $N$  in (c) and  $r$  in (d).

**Percentage of coverage reduction:** Figure 5(c-d) shows the impact on coverage, measured in terms of percentage of coverage reduction relative to the pre-failure level, while changing the  $N$  and  $r$ . The action range is set to 50m in these experiments. Overall, ACR limits the coverage loss and consistently outperforms other approaches. Obviously, the advantage of ACR is due to moving high degree non-critical nodes with high overlapped coverage. Moreover, limited scope of node relocation also limits the coverage loss at the network periphery. Figure 5(d) indicates that the performance of ACR is not much affected compared to DARA because DARA moves least degree nodes that may not have overlapped coverage. Moving critical nodes also trigger successive relocations that cause significant coverage loss at the network periphery.

## 6 Conclusion

We have presented a novel hybrid Application-centric Connectivity Restoration algorithm that factors in application-level concerns in addition to resource optimization while recovering from critical node failures. The proposed ACR algorithm identifies critical actors and designates for them backup actors as part of pre-failure planning to minimize recovery time. It strives to reduce the scope of recovery and incurred overhead by choosing nearby non-critical neighbors as backups. ACR designates highly connected backup nodes with overlapped coverage in order to minimize the impact of critical node failure on coverage and connectivity. In post-failure recovery, it pursues controlled and coordinated actor relocation in order to reorganize the topology and regain the pre-failure strong connectivity. The simulation results have confirmed the effectiveness of ACR compared to contemporary recovery schemes in terms of minimizing recovery time, satisfying application requirements, reducing recovery overhead and limits the impact of the node failure on the coverage and connectivity.

**Acknowledgement.** Imran, Abas and Halabi are supported by the Univ. Teknologi PETRONAS, while Younis' work is supported by the National Science Foundation, award # CNS 1018171.

## References

1. Akyildiz, I.F., Kasimoglu, I.H.: Wireless Sensor and Actor Networks: Research Challenges. *Ad Hoc Networks* 2(4), 351–367 (2004)
2. Abbasi, A., Akkaya, K., Younis, M.: A Distributed Connectivity Restoration Algorithm in Wireless Sensor and Actor Networks. In: 32nd IEEE Conference on Local Computer Networks, Dublin (2007)
3. Younis, M., Lee, S., Gupta, S., Fisher, K.: A Localized Self-Healing Algorithm for Networks of Moveable Sensor Nodes. In: IEEE Global Telecommunications Conference (GLOBECOM), New Orleans (2008)

4. Tamboli, N., Younis, M.: Coverage-Aware Connectivity Restoration in Mobile Sensor Networks. In: IEEE International Conference on Communications (2009)
5. Imran, M., Younis, M., Md Said, A., Hasbullah, H.: Volunteer-Instigated Connectivity Restoration Algorithm for Wireless Sensor and Actor Networks. In: IEEE International Conference on Wireless Communications, Networking and Information Security, Beijing (2010)
6. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less Low-Cost Outdoor Localization for Very Small Devices. *Personal Communications* 7(5), 28–34 (2000)
7. Basu, P., Redi, J.: Movement Control Algorithms for Realization of Fault-Tolerant Ad Hoc Robot Networks. *Network* 18(4), 36–44 (2004)
8. Das, S., Liu, H., Kamath, A., Stojmenovic, I.: Localized Movement Control for Fault Tolerance of Mobile Robot Networks. In: Orozco-Barbosa, L. (ed.) *Wireless Sensor and Actor Networks*, pp. 1–12. Springer, Boston (2007)
9. Imran, M., Younis, M., Md Said, A., Hasbullah, H.: Partitioning Detection and Connectivity Restoration Algorithm for Wireless Sensor and Actor Networks. In: 8th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2010), Hong Kong (2010)
10. Wang, G., Cao, G., La Porta, T., Zhang, W.: Sensor relocation in mobile sensor networks. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies (2005)
11. Abbasi, A., Baroudi, U., Younis, M., Akkaya, K.: C<sup>2</sup>AM: An algorithm for Application-Aware Movement-Assisted Recovery in Wireless Sensor and Actor Networks. In: Proc. of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, ACM, Leipzig (2009)
12. Milenko, J., Stojmenovic, I., Hauspie, M., Simplot-ryl, D.: Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad Hoc Networks. In: Proc. of 3rd Annual IFIP Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net, Bodrum, Turkey, pp. 360–371 (2004)

# Link Quality-Based Channel Selection for Resource Constrained WSNs

Markku Hänninen, Jukka Suhonen,  
Timo D. Hämäläinen, and Marko Hännikäinen

Tampere University of Technology, Department of Computer Systems  
P.O. Box 553, FIN-33101 Tampere, Finland  
{markku.hanninen,jukka.suhonen,  
timo.d.hamalainen,marko.hannikainen}@tut.fi

**Abstract.** Wireless Sensor Networks (WSN) consist of autonomous and intelligent nodes that combine sensing, actuation, and distributed computing with small size and low energy. One of the major issues is overcoming the non-ideality of unreliable real-world wireless links and avoiding interferences. This paper presents a link quality-based lightweight channel selection mechanism that discovers low interference and lightly loaded channels, thus improving latency, reliability, and throughput. The mechanism grades a channel from link reliability and changes the channel on interference. The proposed selection is suitable for resource and energy constrained WSNs as it does not require any extra communication or channel sensing. The channel selection is implemented with a low energy 2.4 GHz multihop mesh WSN using 1 mW transmission power. In practical measurements in a typical office environment with 100 mW WLAN interference, the selection had 96% average link reliability compared to the 84% of randomized channel selection.

**Keywords:** Wireless sensor networks, Channel Selection, Interference Avoidance.

## 1 Introduction

Wireless Sensor Networks (WSN) are an emerging technology consisting of even thousands of fully autonomous tiny nodes, which sense their environment, control actuators, perform distributed computation, and communicate wirelessly with each other. WSNs have applications for example in home, office, health care, military, and industry. This paper concentrates on resource constrained multihop mesh WSNs in which channels and routes are selected autonomously.

Operating environments of WSNs may be very harsh. Wireless links suffer from external interferences caused by other wireless technologies such as Wireless Local Area Networks (WLAN), internal interferences from other nodes in the network, channel fading, and inclement weather [9]. Also asymmetric links and reflection, scattering, and multipathing of radio signals weaken the channel quality. All these factors decrease the performance of WSNs limiting reliability, throughput, latency, and useful communication range between nodes.

The increase of transmission power and retransmissions are necessary but not sufficient techniques to overcome these problems [2,11]. Also, both of them increase internal interference in the network.

Energy efficiency is a vital issue in battery-powered WSNs. The energy consumption in WSN nodes is typically dominated by transceiver circuitry [5]. Therefore, minimizing the radio duty cycle is important for extending the lifetime of battery-powered nodes. Use of disturbed channels should be avoided as it causes retransmissions and needs higher transmission power, both of which increase the energy consumption.

To achieve high performance and energy efficiency, channel management is needed to select high-quality and interference-free channels and to guarantee reliable operation even in interfering environment. In this paper, high-quality refers to a channel with high link reliability. As regionally large networks can experience interferences on different channels in different parts of a network, the selection should be localized.

Both Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) provided by radio transceiver can be used to estimate link quality needed for channel assessment [13]. However, the lowest energy consumption can be achieved with simple radio transceivers without complex hardware based RSSI and LQI mechanisms [6]. Thus, link quality has to be estimated otherwise.

In this paper, we present the design and implementation of a link quality-based lightweight channel selection. The channel selection grades channel quality by monitoring link reliability and changes the channel if its quality is low. A channel grade is defined by combining link reliability to measured or estimated RSSI. The grading averages short-term quality measurements to reliably distinguish static interferences from temporary dynamic interferences, and thus minimize the overhead of frequent channel changes. The mechanism does not increase energy consumption, since it does not require any extra communication or channel sensing. The selection requires that each node can choose its operating channel individually. Packets are transmitted at the channel of the receiving node. This approach applies to clustered WSNs such as IEEE 802.15.4. The selection is targeted at memory and computing power constrained nodes. It does not require hardware LQI support, allowing its implementation on low complexity transceivers.

The channel selection has been implemented into Tampere University of Technology WSN (TUTWSN), which is a low energy, multihop WSN developed for low data rate applications. It comprises prototype platforms and a protocol stack with a clustered Medium Access Control (MAC) protocol and dynamic cost routing. The functionality of the selection has been verified with practical interference tests.

This paper is organized as follows. The related research is discussed in Section 2. The design of channel selection is described in Section 3 and the implementation in Section 4. Section 5 presents an experimental performance evaluation including comparisons to channel hopping, randomized channel selection, and blacklisting with a fixed link reliability threshold. Section 6 concludes the paper.

## 2 Related Research

Interference can be avoided via link or channel blacklisting. Link blacklisting discards neighbor connections that have low reliability while relying on the ability to route around the areas having problems [10]. A link is typically blacklisted when its reliability is below a threshold [8,2]. Authors in [11] combine blacklisting with transmission power control by avoiding links that require high transmission power. In [9], the authors propose blacklist-aided forwarding in which the next hop is determined based on destination and the blacklist. Each packet carries a list of low-quality links encountered along the routing path, which is used to update the blacklist. The drawback of the method is its integration to the routing, which limits its applicability. Also, the available payload decreases. In general, the problem with the link blacklisting is the requirement of dense, well-connected networks to ensure that a route exists.

Channel blacklisting avoids using unreliable channels. The paper [1] presents a lightweight channel selection scheme for WSNs with cognitive MAC protocols. The scheme is based on heuristics and preamble sampling principle, where nodes poll the medium and sense the carrier asynchronously. The strategy has a disadvantage that scanning all the available channels consumes energy even though the scanning can be performed during the periodic channel polling process. The polling approach restricts the applicability of the scheme to Low Power Listening (LPL) MACs.

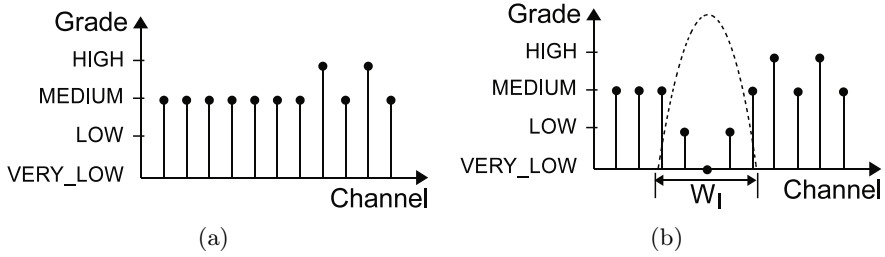
ISA100.11 [3] and WirelessHART [12] utilize channel hopping to minimize the impact of a low-quality channel. However, as channel hopping uses both low- and high-quality channels, the overall performance may not be optimal [10]. ISA100.11 includes also centralized and local channel blacklisting. The centralized blacklisting cannot react to local interferences as it applies to the whole network. The localized blacklisting saves energy by reducing retransmission but increases latency as a node waits until the hop sequence reaches a high-quality channel.

Our approach uses channel blacklisting but does not require channel scanning. Unlike link blacklisting methods, our proposal ensures connectivity by allowing the use of low reliability links when there are no other alternatives.

## 3 Design of Channel Selection

Channel selection uses a reactive approach in which a node keeps its current channel if its quality is deemed satisfactory. The channel can be changed without a priori knowledge of the full channel spectrum usage. This way, the selection avoids the energy consuming scan of channels.

Each channel is assigned with a grade within the set (HIGH, MEDIUM, LOW, VERY\_LOW) indicating the relative quality of a channel, and a time stamp of the last channel assessment moment. The channels are graded by measured link reliability, denoting the probability that a data frame is acknowledged. Thus, the link reliability metric reflects link quality in both directions and there is no need to inject additional traffic to measure reliability. The channel grading is presented in Figure 1.



**Fig. 1.** Channel grading. a) Initial grading as MEDIUM, while reliable links are graded as HIGH. b) An unreliable link is graded as LOW or VERY\_LOW, lowering also the grades of channels within the configured interference range  $W_I$ .

### 3.1 Channel Selection Method

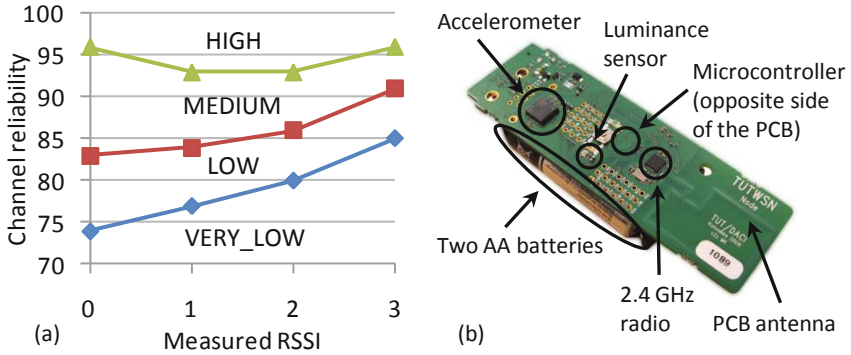
A channel selection is triggered if a node deems its own channel bad (LOW or VERY\_LOW grade). A node selects the highest graded channel available for its operating channel. The selection between the channels having the same grade is based on the time stamp. With HIGH or MEDIUM grades, the most recently updated channel is preferred as it has the highest probability of still being good. With LOW or VERY\_LOW grades, the recently updated channels are avoided as they are most probably unreliable.

A node grades its own operating channel and the channels used by its neighbors with an interval of  $T_U$ . To prevent short-term temporary interferences from triggering a channel change, channel grade is changed only if  $C_W$  consecutive channel updates indicate the same grade. The neighbor channel grading is optional but improves the channel selection as more channels are up-to-date. To avoid overhead, only the neighbors with which a node communicates are considered. As a node actively grades only a subset of channels, the channel information on other channels may become invalid over time e.g. due to the changes in the wireless environment. Therefore, a channel is reset back to the initial value (MEDIUM) after a grade timeout  $T_G$  has elapsed since the last update.

As an interference may be wider than a single channel, the channel grading uses  $W_I$  parameter to denote the expected frequency bandwidth of the interference. When a channel is graded below MEDIUM, also the grades of channels within  $W_I/2$  are lowered. As an example, in 2.4 GHz frequency band,  $W_I$  could be set based on WLAN channel bandwidth to 22 MHz.

### 3.2 Link Quality-Based Channel Grading

Channel grades are configurable and adjusted to the used WSN technology. The MEDIUM grade threshold is critical as it induces a channel change. Depending on the overhead of the channel change on MAC protocol, the channel grade thresholds may be conservative to allow low link reliabilities. Also, the route selection can intentionally use low-quality links. For example, TUTWSN routing may choose a lower reliability link if that minimizes the overall hop count to a routing target, data forwarding energy, and latency in a multihop chain.



**Fig. 2.** Reliability thresholds determined experimentally with TUTWSN. a) Channel grade limits as a function of the measured link RSSI. b) TUTWSN prototype platform.

Another challenge is to identify the situation in which a node does not have any reliable channels e.g. when nodes are deployed too far away from each other, as it can cause unnecessary channel changes. To avoid this situation, the thresholds are defined as a function of RSSI as shown in Figure 2(a). For the implementation, the thresholds were experimentally determined by measuring the typical link reliabilities in TUTWSN platform with four communication ranges corresponding to different RSSI levels. The test nodes sent data to each other and reported link reliability and RSSI values by debug printing. Measured link reliabilities were increased and assigned to the HIGH and reduced and assigned to the MEDIUM grade to allow a slight margin of error and routing behavior. The threshold of LOW was measured in the presence of typical WLAN interference in an office environment. The experimental method offered realistic reliability thresholds with a moderate effort.

## 4 Implementation in TUTWSN Prototype Platform

TUTWSN utilizes beacon synchronized low duty-cycle MAC protocol and multi-cluster tree topology [7]. The channel is accessed in cycles, which consist of a superframe and idle time. For eliminating collisions, superframes have locally unique schedules such that they do not overlap on the same channel. Network density can be increased by utilizing several channels as every node chooses its channel individually. Neighbors are discovered by scanning for network beacons sent on a separate global network channel. This energy optimization does not affect the channel selection as it only allows faster neighbor discovery.

Each superframe begins with a cluster beacon followed by contention and contention-free time slots. The beacon contains slot allocation, cluster status, timing, and routing information. Node allocates the contention-free slots, in which data frames are transmitted, according to the bandwidth needed by associated nodes (members).



In TUTWSN routing data frames are transmitted to a neighbor with a lower cost and a sink has the lowest cost in the network [14]. This kind of routing supports deploying large-scale networks. Routing costs are determined on the basis of link reliability, RSSI and the bandwidth usage of data traffic.

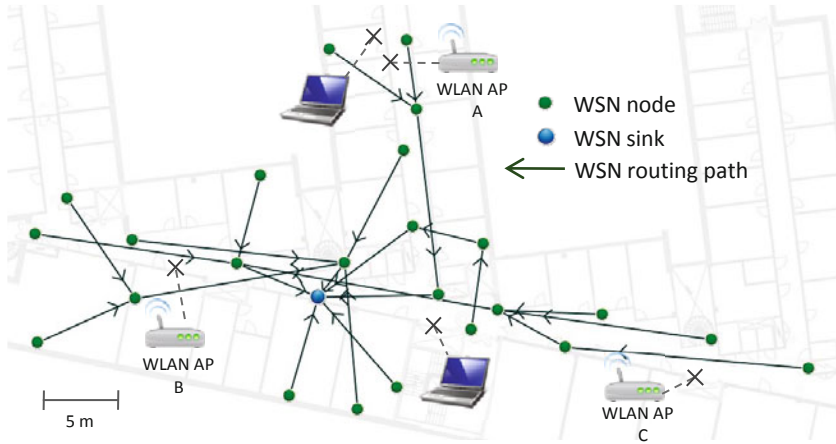
We have fully implemented and integrated the designed channel selection into the TUTWSN protocol stack. The implementation platform presented in Figure 2(b) utilizes a Microchip PIC18LF8722 8-bit microcontroller and 2.4 GHz low-power Nordic Semiconductor nRF24L01 radio transceiver, which is used at 1 Mbps transfer rate with a 32-byte payload. Since the transceiver does not have any RSSI or LQI mechanisms, RSSI is estimated according to succeed and failed receptions of beacon frames transmitted at different transmission power levels as described in [6]. Duplicates of data packets are identified with sequence numbering in MAC headers.

The channel grading interval  $T_U$  was set to 1 min and the grade change threshold  $C_W$  to 3 to assure that enough data traffic is received to assess the channel before a channel change decision. Lower values would improve reactivity but may also cause too rash changes to unfamiliar channels. With greater values the disturbed channel would be kept longer, which decreases the network performance momentarily. The grade timeout ( $T_G$ ) was set from the number of channels  $N_C = 28$  to  $T_U \cdot C_W \cdot N_C = 84$  min to make sure that every channel can be tried before attempting the same channels again. The implementation in C language compiled with MCC18 consumes 1272 bytes program and 44 bytes data memory, which correspond 0.96% and 1.1% of total memory amounts, respectively.

## 5 Performance Evaluation

We have evaluated the performance of the implemented channel selection by interference tests in a typical office environment in two different test scenarios. In a 10-hour static interference scenario, three 802.11g WLAN access points operating statically on channels 1, 6 and 11 generate interference via their normal operation. WLAN access points are significant interference sources due to their high transmission power and wide-band channels compared to WSNs. In these tests, the maximum output powers of WLAN and WSN are 100 mW and 1 mW, respectively. WLAN beacons sent at 1 Mbps utilize 0.744% of channel. From the transferred data amounts logged by the access points and assuming 54 Mbps data rate, we estimate the total average channel utilization below 1.0%.

In a dynamic interference scenario, besides interferences caused by the WLAN access points, two laptops with integrated WLAN adapters are utilized as interference sources using the IEEE 802.11 [4] channels 1, 4, 7, 10 and 13. The duration of a test is five hours and the operating channel is changed once an hour. The laptops are configured to send data with a variable packet length of 40–1200 bytes and 20 ms packet interval to each other in an ad hoc mode. The bandwidth usage of data sent with 11 Mbps data rate is 240 kbps. Including radio, MAC, and UDP/IP headers the total channel utilization is below 7.0%. In both scenarios, the performance is compared with three other channel selection methods: randomized selection, channel hopping, and blacklisting with a fixed reliability threshold. Interference-free environment is considered as a reference level.



**Fig. 3.** Multihop network topology in the interference tests with routing paths and WLAN access points and laptops as interference sources

The evaluation tests have been performed with TUTWSN consisting of 24 nodes and a sink. The network topology with WLAN access points and laptops is shown in Figure 3. The channels are configured from 2.402 GHz to 2.483 GHz at intervals of 3 MHz. 2.480 GHz is reserved for a network channel. Nodes send five data packets per minute and thus total traffic to the sink is 120 packets/min. With 2 s access cycle length and 12 reserved slots/access cycle the maximum throughput of every router node, including the sink, is 360 packets/min.

### 5.1 Compared Channel Selection Methods

In *randomized channel selection*, a node selects randomly one of the free channels available in its time slot. The selection is performed only if a node detects an overlapping time slot with another node operating on the same channel in a two-hop neighborhood.

*Blacklisting with a threshold* is a simpler modification from the selection presented in this paper. A node measures the reliability of its own cluster channel but does not consider RSSI. A channel is blacklisted and a new one selected if the reliability of the current channel decreases below the fixed 85% threshold value.

*Channel hopping* is implemented with 18 different channel sets, each of them containing seven channels. The channel is changed every access cycle. The hopping sequences of different nodes can be timely shifted relative to each other in a WSN neighborhood. In hop sequence design, attention was paid to three main principles. First, channels of two different sequences do not fully overlap even if the sequences were timely shifted. Second, if channels of different sequences are close to each other on one access cycle, they are not on the rest six access cycles. Third, every hop is at least 22 MHz, which is the bandwidth of a 2.4 GHz IEEE 802.11 [4] channel, to minimize the effect of WLAN interference.

**Table 1.** Performance in static interference tests

<b>Performance metric</b>	Interference-free	Randomized selection	Channel hopping	Threshold selection	Link quality-based selection (this paper)
Link reliability (%)					
min	89.0	69.5	73.8	87.2	87.7
avg	96.6	84.3	84.7	94.7	95.9
max	99.9	95.8	91.5	97.1	99.6
Hops avg (count)	2.30	2.91	3.06	2.43	2.44
Latency/hop avg (s)	1.33	1.91	1.74	1.42	1.32
Data forwarding energy avg (%)	100	145	152	108	107

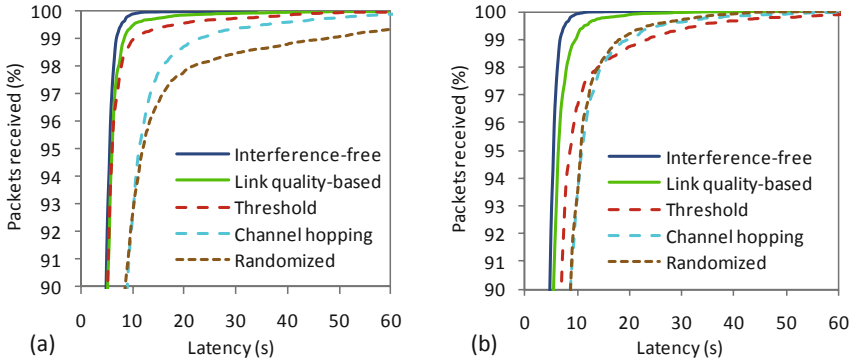
**Table 2.** Performance in dynamic interference tests

<b>Performance metric</b>	Interference-free	Randomized selection	Channel hopping	Threshold selection	Link quality-based selection (this paper)
Link reliability (%)					
min	89.0	71.5	69.9	85.7	81.5
avg	96.6	81.8	81.3	94.5	93.7
max	99.9	94.2	88.5	98.7	99.7
Hops avg (count)	2.30	2.93	2.93	3.01	2.31
Latency/hop avg (s)	1.33	1.69	1.73	1.58	1.51
Data forwarding energy avg (%)	100	150	151	134	104

## 5.2 Measurement Results

The results of the static and the dynamic interference tests are presented in Tables 1 and 2, respectively. The minimum link reliability is the value of the worst and the maximum the value of the best node. Hop count is the number of hops from a source node to the sink. Data forwarding energy is calculated as  $hop\ count(avg) \times 1/link\ reliability(avg)$ . Link quality-based selection succeeds to avoid interferences and increases the average link reliability 12 percentage points in both cases compared to randomized. Not even interference-free reaches 100% link reliability due to physical matters, such as radio wave reflection, multipathing, and attenuation by walls and other obstacles. Also, the distance between communicating nodes may be so long that even the maximum transmission power is not enough to avoid packet errors.

The average end-to-end latencies in static and dynamic interference tests are presented in Figure 4. Link quality-based selection shortens end-to-end latency compared to randomized and channel hopping in three ways. First, the reliability



**Fig. 4.** End-to-end latency in a) static and b) dynamic interference tests, showing the fraction of packets received as a function of latency

of beacon receptions increases and a node has more often a chance to send data. Second, data transmissions succeed more often at the first attempt. Third, hops are longer and less hops are needed to route data from a source node to the sink. The two last observations decrease data forwarding energy. The benefit of channel hopping compared to randomized is minimal because it utilizes both disturbed and interference-free channels all the time.

## 6 Conclusions

The design and implementation of a link quality-based channel selection for WSNs is presented. According to the evaluation, our proposal can be implemented even with radios lacking RSSI and LQI mechanisms. We have carried out experimental performance evaluation of the channel selection in the presence of both static and dynamic interference patterns. The results show that the selection is able to avoid interferences and utilize interference-free channels in an opportunistic manner. Our proposal achieves high performance gains in link reliability and latency over randomized channel selection and channel hopping.

## References

1. Ansari, J., Mähönen, P.: Channel selection in spectrum agile and cognitive mac protocols for wireless sensor networks. In: Proceedings of the 8th ACM International Workshop on Mobility Management and Wireless Access, MobiWac 2010, pp. 83–90. ACM, New York (2010)
2. Gnawali, O., Yarvis, M., Heidemann, J., Govindan, R.: Interaction of retransmission, blacklisting, and routing metrics for reliability in sensor network routing. In: First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), pp. 34–43 (October 2004)
3. ISA: Wireless systems for industrial automation: Process control and related applications (2009), ISA-100.11a-2009

4. ISO/IEC 8802-11, IEEE Std 802.11 Second Edition: Information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications (August 2005)
5. Jin, F., Choi, H.A., Subramaniam, S.: Hardware-aware communication protocols in low energy wireless sensor networks. In: IEEE Military Communications Conference (MILCOM), vol. 1, pp. 676–681 (October 2003)
6. Kohvakka, M., Suhonen, J., Hännikäinen, M., Hämäläinen, T.D.: Transmission power based path loss metering for wireless sensor networks. In: 17th Int'l Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5 (September 2006)
7. Kohvakka, M., Suhonen, J., Hämäläinen, T.D., Hännikäinen, M.: Energy-efficient reservation-based medium access control protocol for wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 20 pages (2010)
8. Liu, T., Kamthe, A., Jiang, L., Cerpa, A.: Performance evaluation of link quality estimation metrics for static multihop wireless sensor networks. In: 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 1–9 (June 2009)
9. Nelakuditi, S., Lee, S., Yu, Y., Wang, J., Zhong, Z., Lu, G.H., Zhang, Z.L.: Blacklist-aided forwarding in static multihop wireless networks. In: 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pp. 252–262 (September 2005)
10. Ortiz, J., Culler, D.: Multichannel reliability assessment in real world wsns. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 162–173. ACM, New York (2010)
11. Son, D., Krishnamachari, B., Heidemann, J.: Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks. In: First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), pp. 289–298 (October 2004)
12. Song, J., et al.: WirelessHART: Applying wireless technology in real-time industrial process control. In: Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 377–386 (2008)
13. Srinivasan, K., Levis, P.: Rssi is under appreciated. In: The Third Workshop on Embedded Networked Sensors (EmNets) (May 2006)
14. Suhonen, J., Kuorilehto, M., Hännikäinen, M., Hämäläinen, T.D.: Cost-aware dynamic routing protocol for wireless sensor networks - design and prototype experiments. In: 17th International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1–5 (September 2006)

# The Target Coverage Problem in Directional Sensor Networks with Rotatable Angles

Chiu-Kuo Liang and Yen-Ting Chen

Department of Computer Science and Information Engineering  
Chung Hua University, Hsinchu, Taiwan 30012, Republic of China  
ckliang@chu.edu.tw

**Abstract.** Directional sensor network is composed of many directional sensor nodes. Unlike conventional omni-directional sensors that always have an omni-angle of sensing range, directional sensors may have a limited angle of sensing range due to technical constraints or cost considerations. Therefore, it is possible that when directional sensor nodes are randomly scattered in the environment, some interested targets cannot be covered due to the limited angle of sensing direction even if the targets are located in the sensing range of sensors. We propose a Maximum Coverage with Rotatable Angles (MCRA) problem in which coverage in terms of the number of targets to be covered is maximized whereas the degree of angles to be rotated is minimized. We present two centralized greedy algorithm solutions for the MCRA problem. Simulation results are presented to apply angle adjustment algorithm to enhance the coverage of the directional sensor network.

**Keywords:** Directional sensor networks, target coverage, rotatable angles.

## 1 Introduction

In recent years, wireless sensor networks have received a lot of attention due to their wide applications in military and civilian operations, such as environmental monitoring [1], [2], battlefield surveillance [3], health care [4], [5] and volcanoes monitoring. In wireless sensor networks, target coverage is a fundamental problem and has been studied by many researchers. Most of the past work is always based on the assumption of omni-directional sensors that has an omni-angle of sensing range. However, there are many kinds of directional sensors, such as video sensors [6], ultrasonic sensors [7] and infrared sensors [5]. The omni-directional sensor node has a circular disk of sensing range. The directional sensor node has smaller sensing area (sector-like area) and sensing angle than the omni-directional one. Although a directional sensor node has multiple orientations, only one orientation can work at the same time. Therefore, targets within the sensing range of a directional sensor but outside of its selected orientation cannot be covered.

There are several ways to extend the sensing capabilities of sensor nodes. One way is to put several directional sensors of same kind on one sensor node, each of which faces to a different direction. One example using this way is in [7], where four pairs of ultrasonic sensors are equipped on a single node to detect ultrasonic signals from

any directions. Another way is to place the sensor node onto a mobile device so that the node can move around. The third way is to equip the sensor node with a device that enables the sensor node to switch or rotate to different directions. In this study, we adopt the third way to make our sensor nodes to face in different directions.

For simplicity, we consider the following assumptions and notations in this paper. In the directional sensor model, the sensing region of each direction of a directional sensor is a sector of the sensing disk centered at the sensor with a sensing radius. When the sensors are randomly deployed, each sensor initially faces to one of its directions. Each sensor node equips exactly one sensor on it. Thus, we do not distinguish the terms sensor and node in the rest of the paper. If a directional sensor faces to a direction, we say that the sensor works in this direction and the direction is the work direction of the sensor. Furthermore, some interested targets with known locations are deployed in the region. When a sensor works in a direction and a target is in the sensing region, we say that the direction of the sensor covers the target.

In directional sensor networks, how to monitor or cover the interested targets is a challenging problem. This is because that a directional sensor has a smaller angle of sensor range than an omni-directional sensor or even does not cover any target when it is deployed. Therefore, we are interesting in improving the coverage of targets for a randomly deployed directional sensor network. Such problem is called the coverage problem in directional sensor network. In general, the goal of the coverage problem is to achieve the best coverage for interested targets.

In order to improve the coverage of targets for a randomly deployed directional sensor network, we assume that each sensor is equipped with a device that enables the sensor to rotate for different sensing directions. Therefore, we are interesting in finding a way for each sensor to rotate with some angles to cover more targets than it is deployed. Since sensor nodes are randomly scattered in the environment replacing or charging battery is very difficult. Therefore, power saving is an important issue in sensor network. In this paper, we are asked to develop a method that can maximize the coverage in term of the number of targets to be covered whereas the total rotated degrees is minimized. The problem is called the Maximum Coverage with Rotatable Angles (MCRA) problem. We present two centralized angle adjustment algorithms, namely the Maximal Rotatable Angles (MRA) scheme and the Maximum Coverage First (MCF) scheme, for the MCRA problem. Simulation results are also presented to show the performance of our proposed angle adjustment algorithms.

The rest of the paper is organized as follows: Section 2 introduces some related literatures dealing with directional sensor networks. In Section 3, the MCRA problem is formally defined and some notations and assumptions are also introduced. In Section 4, our proposed angle adjustment algorithms are presented. In Section 5, some simulation results are presented for showing the performance of our proposed algorithms. Finally, some concluding remarks are given in Section 6.

## 2 Related Works

The coverage rate of sensor network represents the quality of monitoring. According to monitoring objectives, there are many different coverage models that have been discussed both in omni-directional sensor networks and directional sensor networks.

Some researchers study that the area coverage depends on whether the region is covered by sensor nodes, and some studies take into account the strength of regional coverage and network connectivity in omni-directional sensor networks.

The target coverage problem in omni-directional sensor networks is one of the most important issues that have been widely discussed. When a set of target nodes are scattered in an environment, the authors in [8] assumed that each sensor can only cover one target at a time, and built a coverage timetable to maximize the network lifetime. The authors in [9], [10] present methods to organize all sensors into subsets that are activated successively to extend the network lifetime.

In recent years, the coverage problem in directional sensor networks has attracted considerable attention. The authors in [11] presented a directional sensor model in which each sensor is fixed to one direction, and analyzed the coverage rate and connectivity problem for the directional sensor networks. Besides that, the directional sensor model that allows each sensor to work in several directions is provided in [12], [13]. Cai *et al.* proved the problem of finding a cover set named Directional Cover Set problem (DCS) is NP-Complete [14]. In [15] and [16], the authors proposed the centralized and the distributed algorithms independently to use the method of considering the weight of targets so that each sensor can decide a better orientation for reaching higher coverage rate.

### 3 Rotatable Angle for Coverage Problem

In this section, we are going to present some notations, assumptions and the definition for the MCRA problem. Suppose that we have a sensor set  $S = \{s_1, s_2, \dots, s_n\}$  of  $n$  sensors and a target set  $T = \{t_1, t_2, \dots, t_m\}$  of  $m$  targets in the sensor networks. We assume that each sensor has  $w$  directions, and let  $d_{i,j}$  denote the  $j^{\text{th}}$  direction of the  $i^{\text{th}}$  sensor where  $1 \leq i \leq n$  and  $1 \leq j \leq w$ . Let  $D = \{d_{i,j} | 1 \leq i \leq n, 1 \leq j \leq w\}$  denote the set of the directions of all sensors. As we know that, in a directional sensor network, sensor nodes cannot monitor the target node even if it is within its sensing radius. Fig. 1 presents an example to demonstrate the situation. In Fig. 1(a), there are three targets, say  $t_1, t_2$  and  $t_3$ , located within the sensing radius of sensor  $s_1$ . Suppose that after sensor  $s_1$  is deployed, it has chosen direction  $d_{1,1}$  as the working direction. Thus, sensor  $s_1$  can cover only targets  $t_1$  and  $t_2$  but not target  $t_3$ . Suppose that the sensing device is equipped on a sensor node with a rotatable device. Thus, the sensing direction of a sensor node can be rotated with any angle. Therefore, we can rotate the sensor with some angles to cover more targets. For example, in Fig. 1(a), if sensor  $s_1$  rotates its working direction with some angles, it can cover targets  $t_1, t_2$  and  $t_3$  simultaneously, as shown in Fig. 1(b).

Therefore, in this paper, we aim to rotate the angle of the sensing direction of the sensor node to monitor interested target nodes that are located within its sensing radius but outside its orientation. By doing so, the overall target coverage rate will be increased. In order to save energy, our goal is to maximize the coverage rate while minimizing the total rotating degrees.



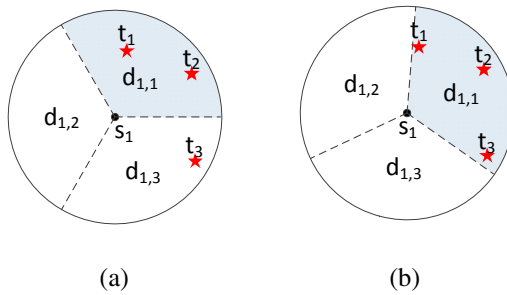


Fig. 1. Simple directional sensor networks with three tunable orientations

## 4 Solutions to MCRA

In this section, we propose two different greedy algorithms to solve the MCRA (Maximum Coverage with Rotatable Angles) problem, that allow directional sensor to rotate its angle to cover as many targets as possible which results in higher coverage in the directional sensor network. In Section 5, we compare the experimental result in term of coverage rate of our algorithms where the directions are rotated, against those methods in [16] where the directions are not rotated.

Section 4.1 describes the weights used in this paper. The detailed solutions for Maximal Rotatable Angle (MRA) scheme and the Maximum Coverage First (MCF) scheme are shown in Section 4.2 and 4.3.

### 4.1 Definition of Weight

In order to cover as many targets as possible, we need to identify, for each target, the number of sensors which can cover the corresponding target. This number can be treated as the weight of targets which can lead us to decide a better way for covering more targets. The weight for each target is called the MCN (Maximally be Covered Number) value, which is equal to the number of effective sensor nodes that can cover the target [16]. An effective sensor node is a sensor node who does not decide its sensing orientation yet. Fig. 2 shows an example. In Fig. 2, there are three effective sensor nodes, namely  $s_1$ ,  $s_2$  and  $s_3$ , and four targets, namely  $t_1$ ,  $t_2$ ,  $t_3$  and  $t_4$ , respectively. As shown in Fig. 2(a), the MCN for  $t_1$  is 3, which means that  $t_1$  is covered by three effective sensors. The MCN values for  $t_2$ ,  $t_3$  and  $t_4$  are 2, 1, and 1, respectively. Note that for our greedy algorithms, the MCN values will be updated once a sensor node decides its sensing orientation. For example, when the sensor node  $s_3$  decides  $d_{3,2}$  as its working direction, it covers target  $t_4$ , but it cannot cover the targets  $t_1$  and  $t_2$ , as shown in Fig. 2(b). Therefore, the MCN values of  $t_1$  and  $t_2$  will be updated as 2 and 1, respectively.

We denote  $w(t_m) = 1/MCN(t_m)$ , as the function of the weight of target  $t_m$ . We also define  $W(d_{i,j}) =$ , as the direction weight of  $d_{i,j}$  where  $d_{i,j}$  denotes the  $j$ -th direction of the sensor node  $s_i$ .

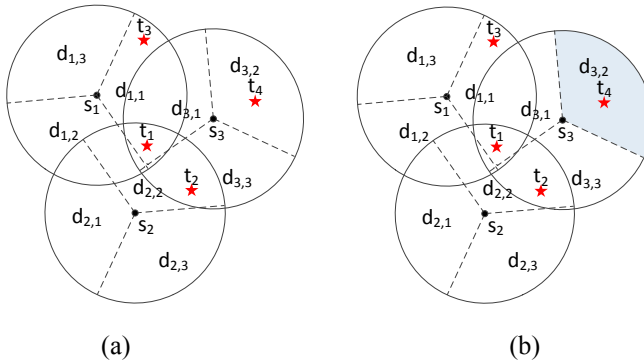


Fig. 2. An example for MCN

### 4.2 Maximal Rotatable Angle algorithm

In this section, we propose a greedy algorithm, called the Maximal Rotatable Angle (MRA) algorithm, which is based on the maximal rotatable angle policy. The idea of maximal rotatable angle policy is that once a sensor was assigned a direction to be its working direction, the sensor could rotate the working direction in order to cover more targets. However, we do not hope the situation that one target (not within the original working direction) can be covered by rotating the working direction with some angles and the other target which is in the original working direction will no longer be covered due to the rotation is occurred. Therefore, our strategy is to keep the targets in the original working direction still in the final working direction after rotating the working direction. Fig. 3 shows the situation. In Fig. 3(a), the working direction of sensor  $s_1$  is  $d_{1,2}$  and there are three targets, say  $t_1$ ,  $t_3$  and  $t_4$ , in the coverage. As we can see in Fig. 3(b), since we need to keep targets  $t_1$ ,  $t_3$  and  $t_4$  in the final coverage, the rotation angle will be  $\alpha_1$  and  $\alpha_2$ , where  $\alpha_1$  and  $\alpha_2$  are the maximal angles of direction  $d_{1,2}$  to rotate clockwise and counterclockwise, respectively.

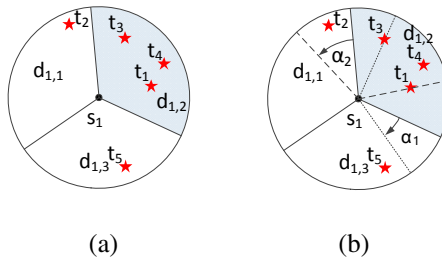


Fig. 3. The rotations of MRA policy

Let  $D$  denote the set of all directions of sensors and  $T$  denote the set of targets. Then, for each target  $t_m \in T$  and each direction  $d_{i,j} \in D$ , our algorithm will first calculate the target weight of  $t_m$  and the orientation weight of  $d_{i,j}$ , respectively. Next, our

algorithm will choose the direction whose orientation weight is maximal among all remaining available directions in  $D$  to be the working direction of the corresponding sensor. Then, our algorithm will apply the MRA policy to the chosen direction to see if it can get a better coverage than the original chosen direction. If there is a better coverage, the direction will be rotated accordingly to have the new working direction; otherwise, the original working direction remains unchanged. The above mentioned procedure will be repeated until remaining available directions is empty or the maximal weight of the chosen orientation is 0.

There is an example to show how our MRA policy works. In Fig. 4(a), the original working direction of sensor  $s_1$  is  $d_{1,2}$  since  $d_{1,2}$  has the maximal weight. After that, the sensor will decide how to rotate to get better coverage than the original coverage. After applying MRA policy, the sensor  $s_1$  will decide to rotate direction  $d_{1,2}$  counter-clockwise to get a new working direction  $d'_{1,2}$ , as shown in Fig. 4(b). As we can see in Fig. 4, there is one more target covered by the new working direction  $d'_{1,2}$  than the original working direction  $d_{1,2}$ .

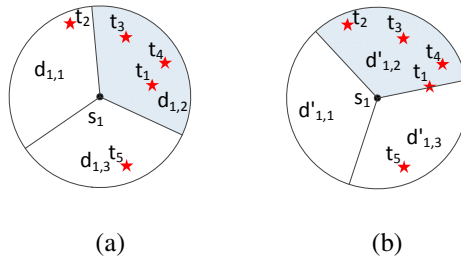


Fig. 4. An example for MRA policy

### 4.3 Maximal Coverage First Algorithm

In this section, we propose another greedy algorithm for MCRA problem. Although we can get a better coverage by applying the previous MRA algorithm, there is a limitation on MRA policy. As we can see in previous section, the original covered targets should be kept in the final coverage regardless the rotation has been applied or not. Thus, the rotation angle is limited and so the more number of targets can be covered. Therefore, we intend to release the limitation in order to cover more targets. This means that we are looking for the rotation that can cover more targets than the original one regardless the rotation angles. Therefore, it is possible that a better coverage can be obtained by uncovering some original targets and covering more new targets. Fig. 5 shows the situation. In Fig. 5(a), the original working direction of sensor  $s_1$  is  $d_{1,2}$  and there are three targets  $t_1$ ,  $t_3$  and  $t_4$  within in  $d_{1,2}$ . However, we can find a better coverage which covers  $t_1$ ,  $t_4$ ,  $t_5$ ,  $t_6$  and  $t_7$  by rotating some angles clockwise as shown in Fig. 5(b). Note that the target  $t_3$  was uncovered in order to cover more targets. The idea is called the Maximal Coverage First (MCF) algorithm.

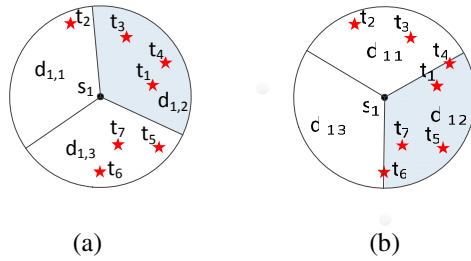


Fig. 5. The rotation scheme of MCF algorithm

Basically, the procedure of MCF algorithm is much the same as in the previous MRA algorithm. The main difference between them is in the rotation scheme. In the MCF algorithm, we do not limit the rotation angles while in MRA algorithm the rotation angles are limited by maximal rotatable angles. Therefore, we will only demonstrate an example to show how the MCF algorithm works and skip the detailed procedures. The detailed procedure of MCF algorithm is omitted.

### 5 Simulation Results

In this section, we will present some simulation results to show the performance of our proposed algorithms. The experimental environment of our simulation is a two-dimensional plane with the size  $R \times R$ , where  $R = 100$  meters. The position of sensor nodes and target nodes are randomly distributed, but any two sensor nodes can not be located in the same position. Each sensor has the same sensing radius which is  $R/10 = 10$  meters, and each sensor node has three directions. All sensor nodes are identical and each sensor node is aware of its own location and can detect target nodes within the sensing range. Any two of the sensor nodes can communicate each other and no communication errors and collisions are occurred. After all sensor nodes and target nodes are spread in the area, all nodes are unable to move. Furthermore, in our experiments, there are 400 targets randomly deployed in the area and the number of sensor nodes is varying from 50 to 225.

Our simulation is designed for the following purposes: evaluating the coverage rate for different approaches, the increasing coverage rate after rotating sensors by unit angle, and the active sensor rate. In the following, we show the simulation results for the above purposes.

First, we present the simulation result of evaluating the coverage rate for different rotating approaches. The Coverage Rate (CR) is used to measure the ratio of target nodes that can be covered in the network. The coverage rate can be calculated by using equation (1), where  $m$ ,  $m_c$ , and  $m_{out}$  represent the total number of targets, the number of covered targets and the number of uncovered targets, respectively.

$$CR = \frac{m_c}{m - m_{out}} \times 100\% \tag{1}$$

It should be noticed that the higher coverage rate is obtained, the better performance of coverage is achieved. Therefore, our purpose is to achieve the higher coverage rate. We evaluate our proposed approaches and compare the performance with previous proposed approach. The experimental result is shown in Fig. 6. In this experiment, we implement five different cases for our proposed MCF (Maximal Coverage First) approach. As mentioned above, each sensor of using MCF approach will rotate its sensing direction and try to cover more target regardless the rotated angles. However, we are interested in finding the benefit of coverage with regard to the rotated angles. Therefore, we implement five different rotating angle limitations to evaluate the performance of coverage. The five different rotating angle limitations are denoted as MCF-10, MCF-30, MCF-60, MCF-90, and MCF-120 which represent the limitation of rotating angles of 10, 30, 60, 90, and 120 degrees, respectively.

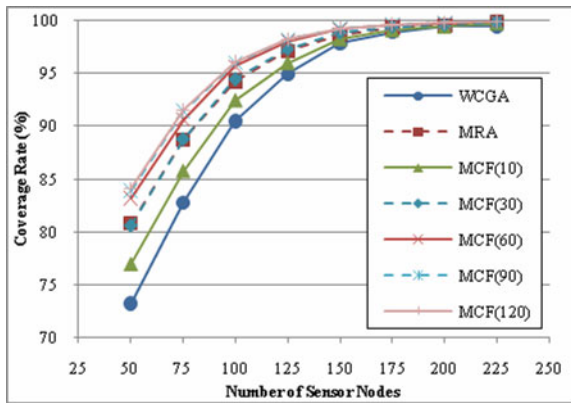


Fig. 6. The coverage rate

As shown in Fig. 6, we can see that our proposed rotating approaches (both MRA and MCF approaches) can achieve the higher coverage rate than the previous proposed WCGA approach [16]. It justifies that we can rotate the sensing direction to get a better coverage rate. It can also be seen that some of the MCF approaches achieve better performance than MRA approach and some of them perform worse. This is because that if we allow the sensor can rotate its sensing direction with a large angle, then it can cover more targets. But if the sensor just rotates its sensing direction with a small angle, then it may not cover any new targets and some of its original covered targets may become uncovered.

The second experiment is focused on the performance of the increasing coverage rate for each sensor after rotating a unit of angle. Fig. 7 shows the experimental results. We can see that the increasing coverage rate of MCF approaches have better performance than that of MRA approach. This means that the rotating scheme in MCF approach can achieve more efficient result than MRA approach in term of the coverage rate.

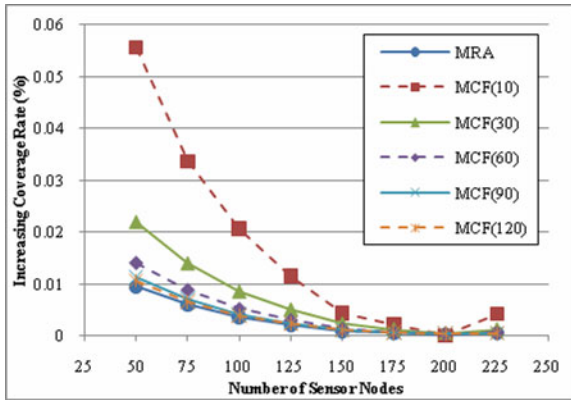


Fig. 7. The increasing coverage rate

The final experiment is focused on the result of active sensor rate. The active sensor rate is the ratio of the number of sensor nodes used for covering the targets to the total number of sensor nodes in the sensor network. The active sensor rate can be used to estimate the power consumption since more sensor nodes being used will consume more electricity. The experimental result is shown in Fig. 8. We can see that both of our proposed approaches, MRA and MCF, achieve better performance than WCGA in term of active sensor rate. This means that our approaches can cover targets by using less sensors than WCGA approach. Therefore, our approach can achieve better performance than WCGA approach in term of energy efficiency.

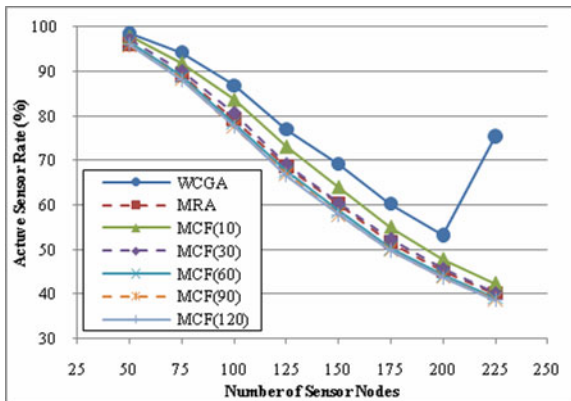


Fig. 8. The active sensor rate

## 6 Conclusions

In this paper, we propose two centralized greedy algorithms, namely the Maximal Rotatable Angle (MRA) scheme and the Maximum Coverage First (MCF) scheme, to increase the coverage rate after the directional sensors are randomly deployed into the

area. Our approaches are using different rotating schemes to increase the coverage rate. Simulation results show that, by rotating the sensing direction of sensors, the coverage rate can be increased effectively compared with the previous WCGA approach. Meanwhile, it can also be found that the MCF scheme can achieve better coverage rate than the MRA scheme. However, the rotating cost of MCF scheme is higher than MRA scheme.

## References

1. Hefeeda, M., Bagheri, M.: Randomized k-coverage algorithms for dense sensor networks. In: *The 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pp. 2376–2380. IEEE Press, Anchorage (2007)
2. Chakrabarty, K., Iyengar, S., Qi, H., Cho, E.: Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers* 51(12), 1448–1453 (2002)
3. Kininmonth, S., Atkinson, I., Bainbridge, S., Woods, G., Gigan, G., Freitas, D.: The Great Barrier Reef Sensor Network. In: *Proceedings of PACEM IN MARIBUS XXXI, Townsville*, pp. 361–369 (2005)
4. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. *ACM Trans. on Multimedia Computing, Communications and Applications*, 102–114 (August 2002)
5. Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., Culler, D.: An analysis of a large scale habitat monitoring application. In: *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 214–226 (2004)
6. Rahimi, M., Baer, R., Iroezzi, O.I., Garcia, J.C., Warrior, J., Estrin, D., Srivastava, M.: Cyclops. In: *situ image sensing and interpretation in wireless sensor networks*. In: *SenSys*, pp. 192–204 (2005)
7. Djughash, J., Singh, S., Kantor, G., Zhang, W.: Range-only slam for robots operating cooperatively with sensor networks. In: *IEEE International Conference on Robotics and Automation* (2006)
8. Liu, H., Wan, P., Yi, C., Jia, X., Makki, S., Niki, P.: Maximal lifetime scheduling in sensor surveillance networks. In: *IEEE INFOCOM* (2005)
9. Cardei, M., Thai, M.T., Li, Y., Wu, W.: Energy-efficient target coverage in wireless sensor networks. In: *IEEE INFOCOM* (2005)
10. Cheng, M.X., Ruan, L., Wu, W.: Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks. In: *IEEE INFOCOM* (2005)
11. Ma, H., Liu, Y.: On coverage problems of directional sensor networks. In: Jia, X., Wu, J., He, Y. (eds.) *MSN 2005*. LNCS, vol. 3794, pp. 721–731. Springer, Heidelberg (2005)
12. Ai, J., Abouzeid, A.A.: Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization* 11(1), 21–41 (2006)
13. Cai, Y., Lou, W., Li, M., Li, X.-Y.: Target-oriented scheduling in directional sensor networks. In: *IEEE INFOCOM*, pp. 1550–1558 (2007)
14. Cai, Y., Lou, W., Li, M., Li, X.-Y.: Energy Efficient Target-Oriented Scheduling in Directional Sensor Networks. *IEEE Transactions on Computers* (TC 2009) 58, 1259–1274 (2009)
15. Cai, Y., Lou, W., Li, M.: Cover set problem in directional sensor networks. In: *Future Generation Communication and Networking (FGCN 2007)*, vol. 1, pp. 274–278 (2007)
16. Chen, U., Chiou, B.-S., Chen, J.-M., Lin, W.: An Adjustable Target Coverage Method in Directional Sensor Networks. In: *IEEE Asia-Pacific Services Computing Conference (APSCC 2008)*, pp. 174–180 (2008)

# UBI-AMI: Real-Time Metering of Energy Consumption at Homes Using Multi-Hop IP-based Wireless Sensor Networks

Timo Ojala<sup>1</sup>, Pauli Närhi<sup>1</sup>, Teemu Leppänen<sup>1</sup>,  
Jani Ylioja<sup>1</sup>, Szymon Sasin<sup>2</sup>, and Zach Shelby<sup>2</sup>

<sup>1</sup> MediaTeam Oulu, University of Oulu, P.O.Box 4500, 90014 University of Oulu, Finland  
{Timo.Ojala, Pauli.Narhi, Teemu.Leppanen, Jani.Ylioja}@ee.oulu.fi

<sup>2</sup> Sensinode Ltd., Hallituskatu 13-17 D, 90100 Oulu, Finland  
{Szymon.Sasin, Zach.Shelby}@sensinode.com

**Abstract.** We present a system for real-time metering of aggregate power consumption and appliance level loads in homes. The power consumption data is collected by sensors in a multi-hop IP-based wireless sensor network. The data is stored at a central server which prepares various representations of the data to be viewed with a web browser. The findings of a longitudinal user evaluation of our system by seven households testify for a successful design and implementation.

**Keywords:** IEEE 802.15.4, 6LoWPAN, wireless embedded Internet.

## 1 Introduction

Despite numerous energy efficiency policies and programs electricity consumption continues to grow. In the EU-25 member states the consumption of the residential sector increased by 10.8% during the period of 1999-2004. In the USA the annual power consumption has tripled during the past two decades. Residential and commercial buildings contribute significant proportion of the total energy consumption, 54% in the EU-25 and 72% in the USA. Further, it has been estimated that 30% of that energy consumption is wasted. [2, 12, 13]

Past studies such as Chetty *et al.* [3] have proven that awareness of energy consumption in homes and offices can lead to up to 20% reduction in electricity usage. While Stern [11] showed a long time ago that real-time, per-appliance data on electricity consumption would provide substantially greater utility and actionable information, the customers of today's utility companies typically have to be content with data that is aggregated, delayed and difficult to access.

Electricity is distributed through a building along a tree-like structure called a *load tree*, where the root connects the building to the power grid and the leaves correspond to individual electric appliances. In an ideal case we would want to have a full, detailed and real-time view of the load tree by monitoring the aggregate load at the root (top-down view) and the individual loads at every appliance (bottom-up view).



However, this is rarely possible in practice due to cost and scalability issues, when a building contains hundreds or thousands of appliances.

Numerous academic and industrial approaches have been proposed for monitoring appliance level loads, for example the MIT Plug [7] for profiling a load over short and long time scales. Many startups have introduced wireless energy monitoring solutions based on ZigBee technology [27]. These products provide detailed power measurements of selected individual loads, which contribute to a bottom-up view of the power consumption.

By monitoring the load at the root a top-down view of the load tree can be pursued. Many utility companies have introduced AMI (Automated Metering Infrastructure) solutions that provide almost real-time data on the aggregate energy consumption of homes. Some utilities are collaborating with aggregators such as Google PowerMeter [18] and Microsoft Ohm [22] to provide more detailed feedback on energy consumption at a household level. Reactive and real power signatures of appliances can be employed for disaggregating individual loads from the aggregate load as originally proposed by Hart [5] and later refined by many studies. The approach is feasible for a small number of loads that have distinguishable differences in power signatures.

We present a system called UBI-AMI for automated power metering of a household by measuring the loads both at the root and at selected individual appliances with wireless sensors. The sensors transmit load measurements over a multi-hop IP-based wireless sensor network (WSN, [10]) to a router which forwards data to a server. The server creates different representations of the data which can be viewed with a web-based user interface (UI) and downloaded as RSS feeds. The UI also allows the user to turn individual sensors (i.e. appliances attached to them) on and off, and request email/SMS alarms based on sensor data. The UBI-AMI system is empirically assessed by a longitudinal user evaluation with seven households.

This paper is organized as follows. The implementation of the UBI-AMI system is presented in Section 2. The setup and main findings of the user evaluation are reported in Section 3. Conclusions and our plans for future work are discussed in Section 4.

## 2 System Implementation

Fig. 1. shows the architecture of the UBI-AMI system, which comprises of three functional building blocks:

1. Sensors in multiple 6LoWPAN sensor networks measure the loads and turn on/off devices connected to the sensors. Each network has a router collecting packets from multiple sensors over a multi-hop mesh topology, i.e. a sensor is able to forward packets of other sensors.
2. Central UBI-AMI server receives packets from the routers, stores the data in a database and creates different representations of the data.
3. User can browse the measurement data with a web browser and subscribe to RSS feeds summarizing the data. Further, the user can control the sensors and subscribe to email/SMS alerts based on load measurements.

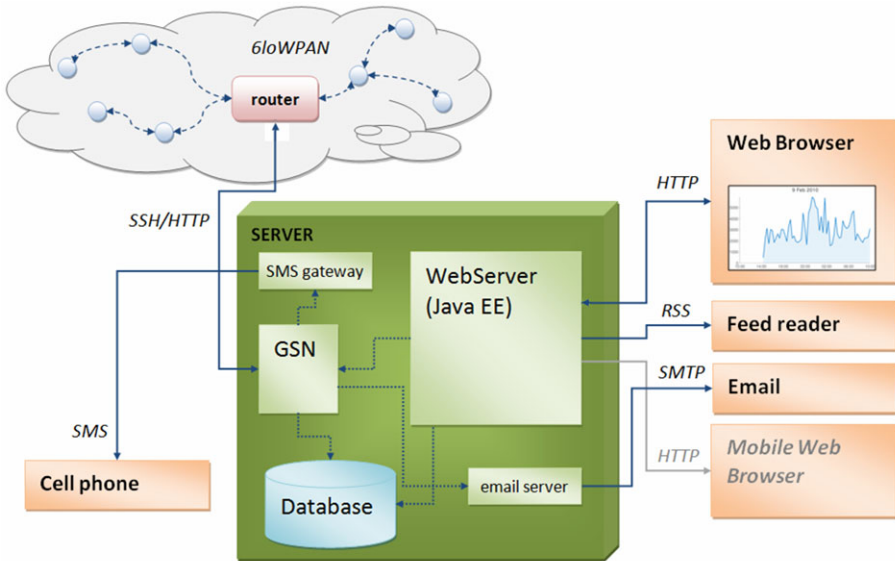


Fig. 1. Architecture of the UBI-AMI system

## 2.1 Sensors

We have built two types of wireless sensors. A *socket sensor* (Fig. 2(a)) measures the loads of individual devices or groups of devices equipped with pluggable power cords. A *mains sensor* (Fig. 2(b)) measures the load at the root of the load tree of a building, including all non-pluggable devices such as electrical heating, hot water boiler and lighting.

The sensors are furnished with a Radiocrafts RC2301 radio module [24] which is equipped with an IEEE 802.15.4 radio on the 2.4 GHz band. The radio module is flashed with the Sensinode NanoStack 2.0 protocol stack [25] (Fig. 3(a)), which implements the IETF 6LoWPAN specification for transmitting IPv6 packets over IEEE 802.15.4 devices [8]. Effectively, the protocol stack provides the sensors with half-duplex multi-hop IPv6 connectivity. The sensors transmit measurement data every 10 seconds and receive possible turn on/off commands in a response message.

### Socket Sensor

The components of a socket sensor are placed on an in-house PCB (Fig. 3(b)), which is isolated from the mains current by an optocoupler and a transformer. The load measurement is conducted with a circuit extracted from a commercial AVEC A9999 energy meter [16]. LM60CIZ temperature sensor [21] and Vishay TEPT5700 illumination sensor [26] are integrated in the enclosure, together with two LEDs providing simple status information.

Atmel ATmega 164P microcontroller [15] collects the data from the energy measurement circuit and the sensors, and communicates the data to the radio module using a socket-like protocol over a serial port. The radio module encapsulates the data

in UDP packets for wireless transmission using the 6LoWPAN protocol over the IEEE 802.15.4 link on the 2.4 GHz band. An external 3 dBi omnidirectional antenna is connected to the radio module to boost the range of the low-power radio.

A relay capable of switching 16 amperes is integrated to turn on/off the appliances plugged into the sensor. The socket sensor is furnished with a 1.5 m three-socket extension cord, to allow easy placement of the sensor unit and the load measurement of multiple appliances simultaneously. The unit price of a socket sensor was 69 EUR when built by us in small quantities.

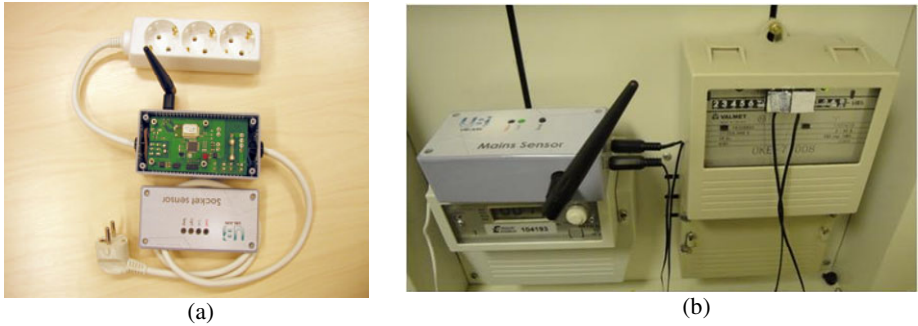


Fig. 2. (a) Socket sensor; (b) Mains sensor

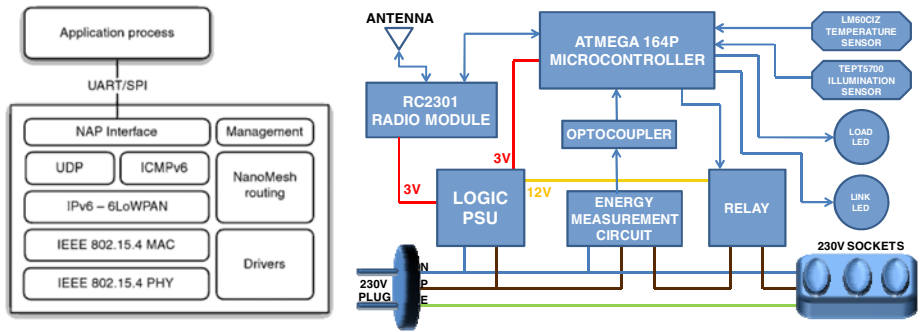


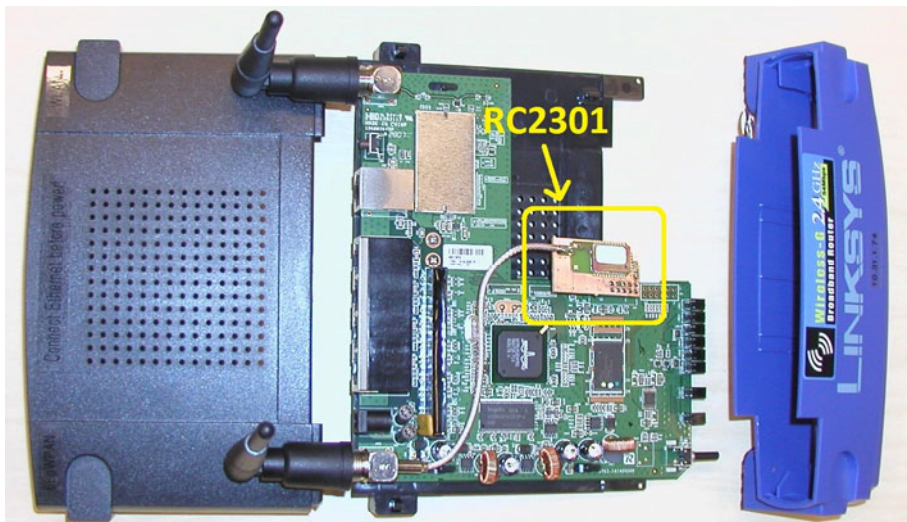
Fig. 3. (a) NanoStack architecture; (b) Block diagram of the in-house PCB

**Mains Sensor**

The mains sensor is built using mostly the same components as in the socket sensor. It measures the load of a building or a home with a simple phototransistor counting the pulses of the S0 interface of a main switch board. The mains sensor does not have mains current on board but is instead equipped with an external power supply. It does not have the extension cord neither the relay unit. The unit price of a mains sensor was 52 EUR when built by us in small quantities.

## 2.2 WSN Router

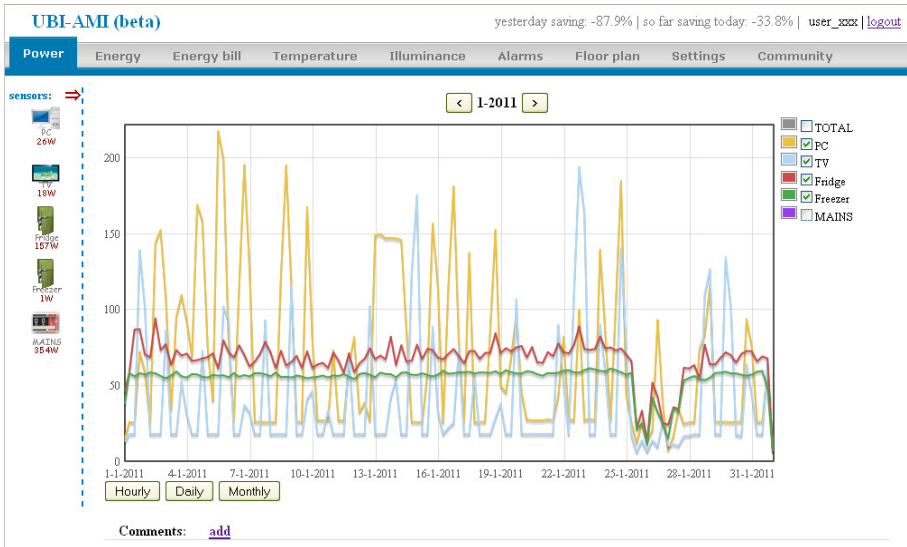
We built the WSN router on an affordable Linksys WRT54GS WLAN router [20] by adding a Radiocrafts RC2301 radio module to a free serial port connected to an external antenna connector (Fig. 4). The Linksys router is flashed with a customized Kamikaze OpenWRT Linux firmware [19] capable of routing packets between the 6LoWPAN network and the Internet. When connected to any IPv4 network providing an IP address via DHCP, the router establishes two SSH connections to the UBI-AMI server. The first connection is used to securely transfer the packets received from the sensors using the HTTP protocol, together with any commands between the router and the server. The second connection is used to access the router remotely for management purposes. The wireless links between the router and the sensors are protected by 128-bit AES encryption. The unit price of a WSN router was 100 EUR when built by us in small quantities.



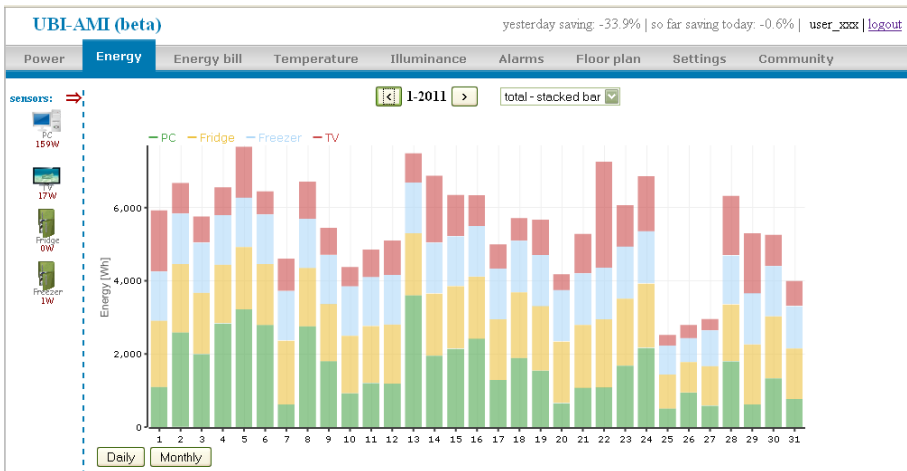
**Fig. 4.** WSN router with the placement of the RC2301 radio module highlighted

## 2.3 UBI-AMI Server and User Interface

The UBI-AMI server is realized in Java atop the open source GSN (Global Sensor Network) middleware in [1]. So-called wrapper components receive the measurement data from the routers and store the data into a MySQL database [23]. For security purposes the server authenticates each router with MAC address filtering and all communication takes place through SSH tunnels. To save storage space, at certain intervals the server compresses and disposes of 'old' data by representing it with average values. The server also keeps tracks of the alarms configured by the users.



(a)



(b)

**Fig. 5.** Screenshots of UBI-AMI user interface: (a) power consumption; (b) energy

The UBI-AMI user interface (UI) is implemented as a website atop the open source GlassFish application server [17] as Java Server Pages. Fig. 5 shows screenshots of the two most commonly used pages, power and energy. Other views based on the sensor data include energy bill, temperature, and illumination. Each type of data can be viewed on different time scales (hourly, daily, weekly, monthly). The UI has separate tabs for drawing the floor plan and for configuring alarms received either via email or SMS to mobile phone. To stimulate participation and community engagement the users can view the consumption data of other members in the community.

The UI has also a separate tab for configuring settings such as the price of energy and RSS feeds reporting the real-time power consumption for the last 1, 4 or 24 hours. The header shows the user's current real-time energy consumption with relative change over the past 24 hours.

When a user starts using the UBI-AMI service, (s)he has to first deploy the sensors around home. Once the floor plan is drawn the icons of sensors can be placed on it and assigned with their friendly names (e.g. 'TV' or 'fridge'). When the icon of a sensor is clicked, a real-time view of the sensor's latest data measurement is provided. Further, in case of a socket sensor the user can turn on/off the sensor, i.e. the appliances plugged into the sensor.

### 3 Longitudinal User Evaluation

The UBI-AMI system was evaluated by seven households in the Oulu region in 2010. Each household was provided with a demo kit comprising of a router, a mains sensor and four socket sensors. The kits were deployed in late January 2010, so that researchers first conducted an initial interview mapping the household's electrical appliances and attitudes towards saving energy. Then the researchers introduced the UBI-AMI system and showed how to install the router, the socket sensors and the mains sensor and how to use the web UI. Quantitative data has been obtained by logging at the server and qualitative data by follow-up interviews of the test users.

All seven households lived in a detached house equipped with electrical heating, an electric sauna and a hot water boiler. While some of the households had previously used the traditional separate energy meters for determining appliance level consumptions, none of them had used any WSN based monitoring systems similar to our UBI-AMI.

Table 1 shows that the test users listed easy deployment, low operational expenses, possibility to measure electricity consumption of an individual appliance and timeliness of the electricity consumption data as the most relevant features of the UBI-AMI service in the opening interviews.

Table 2 tallies the test users' expectations on the impact of the UBI-AMI service on their households. While all test users expected the UBI-AMI service have some sort of an effect, none expected the electricity consumption to reduce a lot.

While the households have used the UBI-AMI system for 11 months, we only use the quantitative data collected in August- December 2010 in this study. By omitting the first six months we wish to eliminate any bias contributed by the curiosity and novelty value of a new system introduced in a household. Thus, we assume that after six months the UBI-AMI system has turned into one of the many services used by a household. During the five month period a total of 219 sessions took place in the UI. Table 3 shows how these sessions were distributed over the different views of the UI in terms of frequency and browsing time. We observe that the power and energy views dominate the usage as expected. While both of them can be viewed at different time scales, over 90% of the viewing takes place at the default daily view.

**Table 1.** Relevance of various features reported by test users in the opening interview

Feature	Count
Low purchase price	5
Low operational expenses	6
Easy deployment	7
Good usability	5
Use of the service via web	5
Use of the service with a mobile phone	2
Configurable email alerts	1
Configurable SMS alerts	4
Possibility to measure electricity consumption of an individual appliance	6
Timeliness of the electricity consumption data	6
Accuracy of the electricity consumption data	2
Automatic archiving of the electricity consumption data	5
Comparison of the electricity consumption data with other users	3
Possibility to turn off/on sensors and attached appliances remotely	3

**Table 2.** Test users' expectations on the impact of the UBI-AMI service

Impact	Count
No effect on the use of electric appliances or electricity consumption	0
We will monitor the use of particular electric appliances more carefully	6
Our electricity consumption will reduce a little	6
Our electricity consumption will reduce a lot	0
Other effect	1

**Table 3.** The distribution of sessions over different views of the UI

View	Frequency (%)	Browsing time (%)
Power	52.4	62.3
Energy	18.5	24.5
Energy bill	13.2	3.7
Temperature	4.4	1.8
Illumination	5.3	3.8
Alarms	1.8	0.1
Floor plan	2.4	2.1
Community	2.0	1.7

In the follow-up interviews the test users were asked to assess various statements on a 5-point Likert scale (1=totally disagree ... 5=totally agree). Table 4 summarizes responses to selected statements so that responses 1 and 2 are tallied under "disagree", 3 under "don't know" and 4 and 5 under "agree". Exactly one test user reported the UBI-AMI service to have had reduced the energy consumption in their household. Overall, the responses testify for successful design and implementation. In some cases the observed technical unreliability of the system was actually attributed to the household's Internet connection being temporarily disconnected.

**Table 4.** Test users' assessment of selected statements

Statement	Disagree	Don't know	Agree
Deployment of socket sensors was easy	0	0	7
Socket sensors functioned reliably	1	1	5
Deployment of mains sensor was easy	1	0	6
Mains sensor functioned reliably	1	1	5
Deployment of router was easy	0	0	7
Router functioned reliably	2	1	4
Web UI was easy to learn to use	0	0	7
Web UI is easy to use	0	1	6
Web UI is clear	1	1	5
Web UI is pleasant to use	1	3	3
Possibility to turn off/on sensors and appliances remotely is useful	1	2	4
Configurable alarms are useful	1	0	6
UBI-AMI is useful service	0	0	7
UBI-AMI gives reliable information on energy consumption	1	1	5
UBI-AMI gives comprehensive information on energy consumption	1	0	6
UBI-AMI gives timely information on energy consumption	0	0	7
UBI-AMI has improved our awareness of energy consumption	0	0	7
UBI-AMI has impacted our way of using electric appliances	3	1	3
UBI-AMI has reduced energy consumption in our household	3	3	1
I wish to use UBI-AMI in the future	0	1	6
I would recommend UBI-AMI to other people	0	0	7

Test users reported that their continuous interest in real-time loads of individual appliances wore out rather quickly. Instead, they would have preferred visualizing the monetary expenses of energy consumption, together with any rapid relative changes, as an ambient media whose observation would not require any conscious or dedicated action on their part. Further, sensors for measuring non-socket appliances such as sauna stoves and hot water boilers were requested, together with automatic incorporation of outdoor temperature, which in Finland is crucial for assessing the contribution of electric heating into the power consumption.

## 4 Conclusion

Our study shows that visibility to both the aggregate load and appliance level loads in a home is very useful functionality that has an impact on how people use electric appliances. IP-based wireless sensor networks and web provide a very potent technology platform for implementing such functionality in a cost-effective, reliable and standardized manner, in contrast to the many proprietary solutions on the market.

While many systems similar to our UBI-AMI have been proposed earlier, e.g. [6, 9, 14], our study is distinguished from most other studies by the longitudinal evaluation of the system by seven real households. Such long-term assessment by real users in authentic setting is needed to reliably study the adoption and impact of this type of ubiquitous computing systems, which require sufficient technical and cultural



readiness to be successful [4]. Further, our system employs multi-hop data dissemination, and allows users to configure alarms based on measurements and to turn on/off appliances remotely.

The UBI-AMI system has also other shortcomings in addition to those discussed in the evaluation. The sensors use unreliable transport protocol (UDP), assuming that a packet is forwarded all the way to the UBI-AMI server after transmission. However, if for example the Internet connection between the router and the server is disconnected, the packets and thus the data are lost. This could be addressed by either the sensors or the router storing the measurements for delay tolerant delivery or by using a reliable transport protocol, but at the cost of extra computation and communication overhead. The 2.4 GHz radio coupled with the 1 mW transmission power does not provide sufficient penetration and range for networking in brick buildings. The design of our sensor casings and the plastic cables limited deployment to dry indoor facilities with nonfreezing temperatures.

We are currently busy building new types of sensors. Their radios will use the sub-GHz (868 MHz) radio, which provides much better penetration and range than 2.4 GHz. We will integrate so-called clamp sensors into the mains sensor for measuring the loads of non-socket appliances from the main switch board. We are also considering other types of sensors such as environmental sensors. We also need to enhance the UI in many respects, e.g. to include budget tracker and energy bill estimation [18] and mobile phone client [14].

The new sensors and the improved UI will be employed in a larger trial to be conducted atop a city-wide wireless sensor network dubbed panOULU WSN. We have deployed around downtown Oulu dozen routers, which provide half-duplex multi-hop connectivity to low power sensors on the sub-GHz (868 MHz) band using the IEEE 802.15.4 radio and the 6LoWPAN protocol stack. The routers provide typically 500 meter line-of-sight range with 1 mW transmission power. Households within the coverage area of the panOULU WSN will not need the dedicated router used in this study.

## Acknowledgments

The authors gratefully acknowledge financial support from the Finnish Funding Agency for Technology and Innovation, the European Regional Development Fund, the City of Oulu, and the UBI (UrBan Interactions) consortium.

## References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: 32nd International Conference on Very Large Databases, pp. 1199–1202 (2006)
2. Bertoldi, P., Atanasiu, B.: Electricity consumption and efficiency trends in the enlarged European Union. European Commission Report EUR 22753 EN (2006)
3. Chetty, M., Tran, D., Grinter, R.E.: Getting to green: understanding resource consumption in the home. In: 10th International Conference on Ubiquitous Computing, pp. 242–251 (2008)

4. Greenberg, S., Buxton, B.: Usability evaluation considered harmful (some of the time). In: 26th Annual CHI Conference on Human Factors in Computing Systems, pp. 111–120 (2008)
5. Hart, G.: Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 1870–1891 (1992)
6. Jiang, X., Van Ly, M., Taneja, J., Dutta, P., Culler, D.: Experiences with a high-fidelity wireless building energy auditing network. In: 7th ACM Conference on Embedded Networked Sensor Systems, pp. 113–126 (2009)
7. Lifton, J., Feldmeier, M., Ono, Y., Lewis, C., Paradiso, J.A.: A platform for ubiquitous sensor deployment in occupational and domestic environments. In: 6th International Conference on Information Processing in Sensor Networks (2007)
8. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: Transmission of IPv6 packets over IEEE 802.15.4 networks. IETF RFC 4944
9. Stavropoulos, T.G., Tsioliariidou, A., Koutitas, G., Vrakas, D., Vlahavas, I.: System architecture for a smart university building. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010. LNCS, vol. 6354, pp. 477–482. Springer, Heidelberg (2010)
10. Shelby, Z., Bormann, C.: 6LoWPAN: The wireless embedded Internet. Wiley, UK (2009)
11. Stern, P.: What psychology knows about energy conservation. *American Psychologist* 47(10), 1224–1232 (1992)
12. U.S. DOE: Buildings Energy Data Book (2008)
13. U.S. DOE: Commercial buildings energy consumption survey (2003)
14. Weiss, M., Mattern, F., Graml, T., Staake, T., Fleisch, E.: Handy feedback: connecting smart meters with mobile phones. In: 8th International Conference on Mobile and Ubiquitous Multimedia (2009)
15. Atmel ATmega 164P, [http://www.atmel.com/dyn/resources/prod\\_documents/doc8011.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc8011.pdf)
16. AVEC A9999, [http://www.semec.fi/product\\_info.php?products\\_id=55](http://www.semec.fi/product_info.php?products_id=55)
17. GlassFish, <http://glassfish.java.net>
18. Google PowerMeter, <http://www.google.org/powermeter>
19. Kamikaze OpenWRT, <http://kamikaze.openwrt.org/7.09/>
20. Linksys WRT54GS, [http://downloads.linksysbycisco.com/downloads/datasheet/1224638902575/wrt54gs\\_ds\\_2.pdf](http://downloads.linksysbycisco.com/downloads/datasheet/1224638902575/wrt54gs_ds_2.pdf)
21. LM60CIZ, <http://www.national.com/ds/LM/LM60.pdf>
22. Microsoft Hohm, <http://www.microsoft-hohm.com>
23. MySQL, <http://www.mysql.com>
24. Radiocrafts RC2301, [http://radiocrafts.com/uploads/rc230x\\_data\\_sheet\\_1\\_2.pdf](http://radiocrafts.com/uploads/rc230x_data_sheet_1_2.pdf)
25. Sensinode NanoStack 2.0, <http://www.sensinode.com/EN/products/software.html>
26. Vishay TEPT5700, <http://www.vishay.com/docs/81321/tept5700.pdf>
27. ZigBee, <http://www.zigbee.org>

# An Accurate and Self-correcting Localization Algorithm for UWSN Using Anchor Nodes of Varying Communication Range

Manas Kumar Mishra<sup>1</sup>, Neha Ojha<sup>2,\*</sup>, and M.M. Gore<sup>1</sup>

<sup>1</sup> Department of Computer Science & Engineering  
Motilal Nehru National Institute of Technology, Allahabad, India  
(manasmishra,gore)[@mnmit.ac.in](mailto:mnmit.ac.in)

<sup>2</sup> Wipro Technologies, Bangalore, India  
[neha.ojha@wipro.com](mailto:neha.ojha@wipro.com)

**Abstract.** Precise event detection in underwater wireless sensor network requires the position information of the detecting sensor nodes. Manual positioning of nodes in such a network is highly infeasible. The popular dive and rise method used in underwater sensor network contributes to the inaccuracies in position estimation by localization methods. Therefore, localization techniques need to have the provision for error correction to improve the accuracy of position estimations. This paper presents a range-free localization algorithm using dive and rise with anchor nodes of varying communication range. The proposed approach uses different error correction methods based on geometric principles to improve upon the accuracy. The simulation results show the effectiveness of the proposed error correction methods on the estimated positions.

## 1 Introduction

Underwater Wireless Sensor Network (**UWSN**) has been suitable for various critical applications like study of underwater ecosystem, pollution monitoring, early warning systems for natural disasters like tsunamis, oil drilling, etc. These applications model the UWSN to operate in a three dimensional (**3D**) space, as the event detection needs to be associated with the depth of its occurrence as well. Preciseness in the estimation of the location of an event is based on the accuracy of the location information of the participating nodes, detecting that particular event. So, it is desired that the nodes are not only aware of their location, but, are also sure of its accuracy. Therefore, the challenge is to come up with an algorithm which helps participating sensor nodes to self-localize with accurate estimations. In addition to this, the harsh physical environment of UWSN makes localization of sensor nodes further challenging. The most widely used localization technique of Global Positioning System (**GPS**) does not work underwater [1]. Apart from this, acoustic signals are used in UWSN instead of

---

\* This work was carried out at Motilal Nehru National Institute of Technology, Allahabad during her stay there as a Master's student.

radio waves [7]. Therefore, several localization techniques [9,10,11,14] developed for terrestrial networks can not be directly applied to UWSN. As GPS does not work underwater, the popular Dive and Rise (DNR) is used by the anchor nodes to know their position information [4]. The mobile anchor nodes repeatedly broadcast their position information, the static nodes register valid beacon points from these messages. Out of the two possible valid beacon points, one is recorded when the static node is at the surface of the communication sphere of the anchor node and before entering in to it, and the second one is recorded at the surface of the communication sphere again when the node is about to leave it. For anchor nodes using DNR with constant velocity as the mobility model, the positions of a set of anchor nodes lie in a plane if their positions are measured with the same time difference. Such positions result in a zero determinant which forces the static nodes to drop the position records which increases the localization time. Therefore, the use of anchor nodes with different communication ranges will be more efficient. Further, the benefits of using anchor nodes with varying communication range have been advocated and proved in [10]. But, the authors are unaware of any range-free scheme which uses anchor nodes with variable communication ranges for position estimation in UWSN. In addition to this, the estimated position will inherit the inaccuracies in readings from the beacon points, if any, due to faultier GPS and pressure gauge readings, wrongly estimated communication ranges, etc. This paper proposes an accurate and self-correcting localization algorithm for UWSN using anchor nodes with variable communication range based on the approach proposed in [10].

## 2 Related Work

Localization schemes that use reference nodes can be broadly classified into range-based and range-free schemes [5]. Range-based schemes that use TOA or TDOA with RF signals for range estimations require perfect time synchronization among the nodes which is difficult to achieve in UWSN [3,4,15] as RF waves do not propagate well underwater. Acoustic signals are used, but, the speed of sound is again affected by the varying temperature, salinity and depth. The resulting estimated positions from these schemes are, thus, not very accurate. RSSI schemes are also not preferred due to surface reflection, bottom reflection and back scattering [3].

In [6] Tian et al. addressed a scheme for underwater acoustic sensor network. This scheme localizes the sensor nodes deployed over three dimensional underwater volume with time synchronization. This space is partitioned into equal sized non-overlapping cells of truncated octahedron in shape. The scheme uses multilateration and acoustic ranging techniques for localization. A set of anchor nodes are placed at the surface of water, and the nearby unlocalized nodes get localized and synchronized with the help of these anchor nodes. These newly localized nodes now behave as anchor nodes and broadcast new synchronization packets. The process gets repeated tier to tier. The major drawback with this approach is that, the localization error at any level or tier is added in further iterations. The error reaches to a very high value if the network goes beyond the fifth tier.

Chandrashekhar and Seah in [2] proposed a centralized range-free 2D area localization scheme for UWSN in which a coarse estimation of the node location is done using multi-power transmission of acoustic signals. Yi Zhou et al. extended this scheme for 3D space in [16] utilizing Detachable Elevator Transceivers for broadcasting reference positions.

In [10] Mishra and Gore proposed a robust localization approach for 3D WSN with error correction. But, as GPS does not work underwater, we used the approach for position calculation and error correction of [10] in this work, and modified the mobility model to adapt to 3D UWSN.

### 3 Preliminaries

#### 3.1 Beacon Point Selection

As the anchor nodes DNR, they repeatedly broadcast their positions and their current communication range. The static nodes listen to all these messages, but store only the first and last messages received from the passing anchor node. Because, the instance at which the static node receives first/last message from an anchor, it is assumed to be at the surface of the communication sphere of that anchor node. The distance between the static node and the anchor is equal to the current communication range of that anchor node. So, it stores the position coordinates and communication range of the anchor node received in the message and considers the position coordinates as a valid beacon point. Figure 1 shows

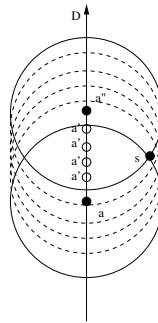


Fig. 1. Valid beacon points  $a$  and  $a''$  received by  $s$

the valid beacon points of  $a$  and  $a''$  recorded by the static node  $s$  for an anchor node moving in  $D$  direction. All other beacon points denoted as  $a$  are not valid as  $s$  is within the communication range of the anchor node rather than being at the surface of it. If  $d$  is the distance of separation between the lines containing the anchor and the static node, respectively, and if the communication range of the anchor node is  $r$  and the speed of the anchor node is  $v$ , then the time difference between the two valid beacon points denoted as  $t$  is given by the equation,

$$t = \frac{2 * \sqrt{r^2 - d^2}}{v} \tag{1}$$

Therefore, for a predefined speed of DNR and a specific distance of separation between a pair of static and anchor nodes, the communication range of the anchor node governs, and is directly proportional to the time difference between the record of two valid beacon points that can be generated from that anchor node.

### 3.2 Position Calculation

When any sensor node registers a valid beacon point, then the sensor node can be assumed to be at the center of a sphere with a radius that denotes the distance of it from the anchor node, while the anchor node can be assume to be at the surface of the sphere. The equation of such a sphere centered at the point  $(x, y, z)$  representing the position of the sensor node and having the anchor node at  $(x_0, y_0, z_0)$  on the surface with radius  $r$  (which is equal to the transmission range of the anchor node) is given by

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \quad (2)$$

Four such equations representing four valid beacon points can be used to determine the position of the corresponding node. The details of the position calculations are adopted from [10].

### 3.3 Error Detection and Correction

The major contribution of this approach is the error correction aspect of the algorithm which provides the much needed robustness to it. Once the location information of a node is estimated, it goes to a state of estimated position. On receiving a fifth valid beacon point it estimates the Euclidean distance between it's estimated position and the position information of the beacon, and compares it with the communication range received. If the distance and the communication range are found to be different, then the error correction is to be performed. The error correction can be achieved by using any of the three different approaches proposed in [10].

## 4 Proposed Approach

The assumptions made for modeling the UWSN for this work is presented in Table 1.

### 4.1 Localization Algorithm

The proposed method uses the GPS enabled anchor nodes deployed in a grid pattern to know their position at the surface of the water, and use DNR with constant velocity to send beacons regarding their position estimations. The anchor nodes are equipped with pressure sensors which help them to estimate their depth underwater. The position of an unlocalized sensor node is calculated based on the position information in the messages received from the anchor nodes. Algorithm 1 presents the detailed algorithm for localization of the static sensor nodes.

**Table 1.** Network Model and Assumptions

Types of Node	Sensor Nodes (large in number) Anchor Nodes (few in number).
Embedded Hardware	Sensor Nodes equipped with sensing purpose hardware. Anchor Nodes are equipped with GPS device to detect their position at the surface of water, and pressure sensors to find their depth underwater.
Battery Power	Anchor Nodes have more battery power as compared to Sensor Nodes as they have to continuously transmit the beacon messages.
Spreading shape of Communication Range	Spherical for all the nodes
Communication Range	Fixed Range (Sensor Nodes) Variable Range(Anchor Nodes). The anchor nodes can have different communication ranges to start with which will remain fixed till completion of the localization process, or may reduce their communication range with an initial fixed value which is same for all anchor nodes, or may have both the conditions. Under all such conditions, the anchor nodes are assumed to know their current transmitting power and are able to calculate the spreading distance of the acoustic signals which they can transmit with that power. These calculated spreading distances are transmitted in their beacon messages as their respective communication range.
Node Mobility	The sensor nodes are assumed to be static in nature unless some external force makes them to change their location. The anchor nodes are considered to be mobile using the mobility model of DNR.

**Algorithm 1.** Localization Algorithm

---

```

Initialize:  $k = 0$  and  $loc\_found = false$ 
  if  $pos\_recvd, range\_recvd$  then
    if  $k < 4$  then
       $add\_pos(SET, pos\_recvd, range\_recvd)$ 
       $k = k + 1$ 
    else
      if  $k == 4$  then
        if  $!loc\_found$  then
           $det = calculate\_det(SET)$ 
          if  $det > 0$  then
             $pos\_est = calculate\_pos(SET)$ 
             $loc\_found = true$ 
          else
             $remove\_last\_pos(SET)$ 
             $k = k - 1$ 
          end if
        else
           $dist = calculate\_dist(pos\_est, pos\_recvd)$ 
          if  $dist = range\_recvd$  then
             $sleep(\Delta_R)$ 
          else
            do error correction
          end if
        end if
      end if
    end if
  end if
end if

```

---

Initially,  $k$ , the number of beacon points received by a static node, is set to zero and  $loc\_found$  is set to false. To start with, each static node waits for the beacon messages. With each beacon message, it receives a new position and the

transmission range of the sender anchor node. Each anchor node position that satisfies the boundary condition is added to the SET of node positions. A static node needs at least four node positions in order to calculate its location ( $k=4$ ). After getting these positions, it calculates the determinant. A zero determinant shows that all the four positions lie in a plane hence, the node removes the last recorded position from the SET, and again waits for the beacon messages to get a new valid beacon position. If the determinant is not equal to zero, then its location can be estimated using the position calculations mentioned in section 3.2. After the location is found, the node goes to a state of estimated position and waits for a new valid beacon position to verify the estimated location. On receiving a fifth valid beacon point, the distance between this position and the estimated location is found. If this distance is equal to the transmission range of the anchor node, then the estimated location is assumed to be correct, otherwise the error correction is done using either of the methods mentioned in [10]. The algorithm terminates once all the nodes are localized.

## 5 Simulation Results and Analysis

The proposed approach is simulated on Sinalgo-0.75.3-Regular Release [12]. It provides a simulation framework for testing and validating network algorithms in both two and three dimensions. Table 2 presents the simulation settings applied for the execution of the proposed algorithm.

**Table 2.** Simulation Settings

Total Number of Sensor Nodes	525
Static Sensor Nodes	500
Mobile Sensor Nodes	25
Simulation Volume:	(500mX500mX500m)
Simulation:	
Synchronous Mode	True
Interference Model	No Interference
Connectivity Model	UDG (rmax=175,150,125)
Reliability Model	Reliable Delivery
Distribution Model	Random(Static Nodes) and Grid(Mobile Nodes)
Mobility Model	No Mobility(Static Nodes), DNR in Grid(Mobile Nodes)
Message Transmission Model	Constant Time
Mobility Model:	
Speed Distribution (meters per round)	Constant(value=1)
WaitingTimeDistribution (number of rounds)	Constant(value=1)
Message Transmission Model:	
Message Transmission Time (number of rounds)	Constant Time(value=1)

The anchor nodes are deployed in a grid (500m X 500m) having 25 squares, each having a side of length 100m. To achieve a connectivity of 4 to 8 anchor nodes from each point in the grid, we have considered three communication ranges for the anchor nodes. The first set of values range from 100m to  $100\sqrt{2}$ m, the second set of values range from  $100\sqrt{2}$ m to  $\frac{3}{2} * 100$ m, and the third set of values range from  $\frac{3}{2} * 100$ m to  $2 * 100$ m. In this simulation, we have considered 125m, 150m, and 175m as the communication ranges. The anchor nodes having same communication range are arranged in a diagonal manner. The grid points have anchor nodes



(in both the dimensions) with communication ranges in an alternating fashion from the three values. So, the corner points in the grid have at least three out of the four connected anchor nodes, having different communication range.

The following subsections present and analyze the results obtained from simulation of the proposed localization algorithm in Sinalgo environment. Two parameters have been used to present and analyze our results with the results obtained using the method proposed by Vijay Chandrasekhar and Winston Seah [2].

- *Average localization time*: The average time required for all sensor nodes to compute their locations i.e.

$$AverageLocalizationTime = \frac{\Sigma Localization\ time}{no.of\ sensors} \tag{3}$$

It provides the rate of localization for any localization approach.

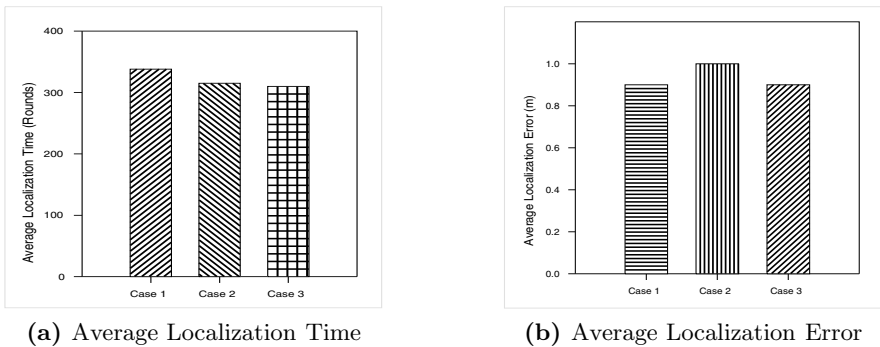
- *Average localization error*: The average distance between the estimated location  $(x_e, y_e, z_e)$  and actual location  $(x_L, y_L, z_L)$  i.e.

$$AverageLocationError = \frac{\Sigma \sqrt{(x_e - x_L)^2 + (y_e - y_L)^2 + (z_e - z_L)^2}}{no.of\ sensors} \tag{4}$$

It checks the accuracy of the computed locations.

### 5.1 Error Correction Methods

The results using anchor nodes of variable communication range have been compared for all the three cases implementing the three error correction methods mentioned in [10], respectively. Figure 2 shows the results. Figure 2a presents the average localization time for all the three error correction methods. Case2 takes relatively less time as discarding all the beacon points makes the method slow in case1, whereas the calculations to be made to determine the candidate



**Fig. 2.** Performance for a sample run of 500 rounds using variable communication range and different error correction methods

beacon point to discard in case3 costs more time. The results in Figure 2b show that, the accuracy in localization is best in case1, whereas, the results in case3 is better than case2. But, the first method of error correction (case1) gains in accuracy at the cost of time of localization.

### 5.2 Average Localization Time

The average localization time as defined earlier is used to compare the results obtained from the approach in [2] with that of the proposed approach. We conducted simulations with 250 static nodes to find out the total localization time in terms of rounds, using both the approaches. Figure 3 presents the results between proposed approach and approach in [2] with 25 anchor nodes being deployed, and using the third error correction method. Further, a set of five readings with 500 static nodes have been taken for each approach to calculate the average localization time for a run of 500 rounds. Figure 4a presents the average localization time for both the approaches.

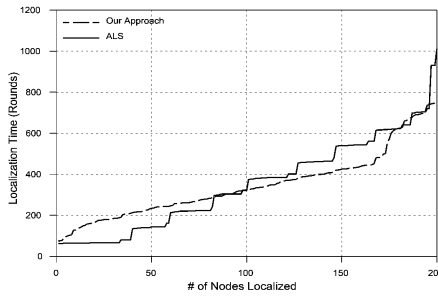
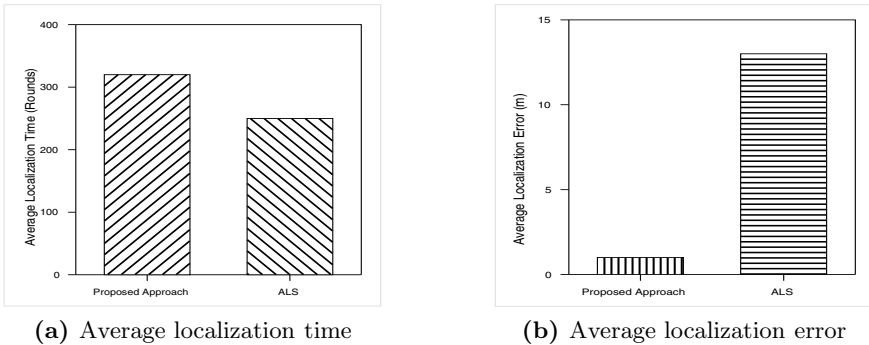


Fig. 3. Performance of different methods with 25 anchor nodes and 250 static nodes



(a) Average localization time

(b) Average localization error

Fig. 4. Performance of different methods for a sample run of 500 rounds

### 5.3 Robustness (Average Localization Error)

The average localization error as defined earlier is also used to compare the results obtained from both the approaches. Figure 4b presents the average localization error for the approach in [2] with the proposed method using the third error correction approach. The results in Figure 4 show that, though the proposed approach takes more time to get the nodes localized, but, its accuracy of localization is a major advantage.

## 6 Conclusion and Future Work

The presented scheme not only localizes sensor nodes but, also verifies the accuracy of the estimated locations. Displaced nodes can relocate themselves using the first error correction method. This scheme requires more beacon messages to be transmitted, but, simple implementation and the accuracy of the estimated locations make the scheme a much better option. The adaptability of the method for the variable communication range makes it more suitable for deployments leading to patchy regions. The error correction methods are driven by the accuracy of the fifth beacon point. We are working on an approach to categorize the GPS readings and provide a weighted measure based on their accuracy so that the impact of the fifth beacon point in verification of the estimated location can be regulated. Determination of an optimal tolerance limit for the mobility of the static nodes without compromising the desired accuracy of location estimation is also planned as a future work.

## References

1. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater Acoustic Sensor Networks: Research Challenges. Elsevier's J. Ad Hoc Networks 3(3), 257–279 (2005)
2. Chandrasekhar, V., Seah, W.: An Area Localization Scheme for Underwater Wireless Sensor Networks. In: Proc. of the IEEE OCEANS Asia Pacific Conference, May 16-19 (2006)
3. Chandrasekhar, V., Seah, W.K.G., Choo, Y.S., Ee, H.W.: Localization in Underwater Sensor Networks - Survey and Challenges. In: Proc. of the International Conference on Mobile Computing and Networking, pp. 33–40 (2006)
4. Erol, M., Viera Luiz, F.M., Gerla, M.: Localization with Dive'N' Rise (DNR) Beacons for Underwater Acoustic Sensor Networks. In: Proc. of the Second Workshop on Underwater Networks, pp. 97–100 (2007)
5. He, T., Stoleru, R., Stankovic John, A.: Range free localization. Technical report, University of Virginia (2006)
6. Jin, J., Wang, Y., Tian, C., Liu, W.-Y., Mo, Y.: Localization and synchronization for 3D underwater acoustic sensor networks. In: Indulska, J., Ma, J., Yang, L.T., Ungerer, T., Cao, J. (eds.) UIC 2007. LNCS, vol. 4611, pp. 622–631. Springer, Heidelberg (2007)
7. Lurton, X.: An Introduction to Underwater Acoustics - Principles and Applications. Springer, Praxis Publishing (2002)

8. Mao, G., Fidan, B., Anderson Brian, D.O.: Wireless sensor network localization techniques. *Computer Network: The International Journal of Computer and Telecommunications Networking* 51, 2529–2553 (2007)
9. Martins, M., So, H.C., Chen, H., Huang, P., Sezaki, K.: Novel centroid localization algorithm for three-dimensional wireless sensor networks. *IEEE Transactions on Wireless Communication, Networking and Mobile Computing*, 1–4 (October 2008)
10. Mishra, M.K., Gore, M.M.: Robust and Distributed Range-Free Localization using Anchor Nodes with Varying Communication Range for Three Dimensional Wireless Sensor Networks. In: Natarajan, R., Ojo, A. (eds.) *ICDCIT 2011*. LNCS, vol. 6536, pp. 209–220. Springer, Heidelberg (2011)
11. Ou, C.-H., Ssu, K.-F.: Sensor Position Determination with Flying Anchors in Three-Dimensional Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 1084–1097 (2008)
12. Sinalgo : Simulator for network algorithms (2009), <http://www.dcg.ethz.ch/projects/sinalgo>
13. Teymorian Amin, Y., Cheng, W., Ma, L., Cheng, X., Lu, X., Lu, Z.: 3D Underwater Sensor Network Localization. *IEEE Transactions on Mobile Computing*, 1610–1621 (2009)
14. Vivekanandan, V., Wong Vincent, W.S.: Concentric Anchor-Beacons (CAB) Localization for Wireless Sensor Networks. *IEEE Transactions on Vehicular Technology*, 2733–2744 (2007)
15. Zhou, Y., Chen, K., He, J., Chen, J., Liang, A.: A Hierarchical Localization Scheme for Large Scale Underwater Wireless Sensor Networks. In: *Proc. of the 11th IEEE International Conference on High Performance Computing and Communications*, pp. 470–475 (2009)
16. Zhou, Y., He, J., Chen, K., Chen, J., Liang, A.: An Area Localization Scheme for Large Scale Underwater Wireless Sensor Networks. In: *Proc. of the 2009 WRI International Conference on Communications and Mobile Computing*, pp. 543–547 (2009)

# Author Index

- Aaltonen, Timo 168  
Abdennhader, Nabil 188  
Ameziani, Hamid 221  
  
Ben Belgacem, Mohamed 188  
Benyettou, Mohamed 117  
Bogunovic, Nikola 3  
  
Cam, Hakan 233  
Chaudhary, Banshi Dhar 62  
Chen, Yen-Ting 264  
Cheng, Feng 138  
Chin, SungHo 13  
Claeys, Laurence 94  
Couturier, Raphaël 188  
Criel, Johan 94  
  
Ermes, Miikka 84  
  
Fan, Xinjin 33  
Ferreira, Paulo 72  
  
Geerts, Marjan 94  
Georgantas, Nikolaos 221  
Gore, M.M. 285  
Green, Daron G. 1  
Grudenic, Igor 3  
Guo, Minyi 42  
Guo, Song 42  
Gustedt, Jens 117  
  
Hämäläinen, Timo D. 254  
Han, Fangfang 33  
Hännikäinen, Marko 254  
Hänninen, Markku 254  
Hasbullah, Halabi 243  
Heikkinen, Arto 158  
Hernane, Soumeya Leila 117  
Hosio, Simo 198  
  
Imran, Muhammad 243  
Issarny, Valérie 209, 221  
Iwata, Takahiro 105  
  
Jin, Hai 42, 52  
Jurmu, Marko 198  
  
Kawsar, Fahim 94  
Kong, Shengyuan 33  
Könönen, Ville 84  
Kristensen, Lars M. 127  
Kukka, Hannu 198  
Kuusijärvi, Jarkko 148  
  
Laitakari, Juhani 178  
Laiymani, David 188  
Lämsä, Arttu 84  
Lee, JongHyuk 13  
Leppänen, Teemu 274  
Li, Jiandun 33  
Li, Qing 33  
Liang, Chiu-Kuo 264  
Liao, Xiaofei 52  
Liikka, Jussi 84  
Lim, JongBeom 13  
Liu, Zhixiang 33  
Lu, Yanchao 42  
  
Mäntyjärvi, Jani 84  
Meinel, Christoph 138  
Mikkonen, Tommi 168  
Miquée, Sébastien 188  
Mishra, Manas Kumar 285  
  
Najafzadeh, Mahsa 23  
Nakajima, Tatsuo 105  
Närhi, Pauli 274  
Niinimäki, Marko 188  
  
Ojala, Timo 274  
Ojha, Neha 285  
Oliveira, Pedro 72  
Ovaska, Eila 148  
  
Pakkala, Daniel 178  
Paloheimo, Harri 84  
Pandey, Mayank 62  
Pantsar-Syväniemi, Susanna 148  
Pathak, Animesh 209, 221  
Peng, Junjie 33  
Perälä, Juho 178

- Qiu, Fei 52
- Rahaman, Mohammad Ashiqur 221
- Rantalainen, Timo 84
- Rautiainen, Mika 158
- Riekki, Jukka 42, 198
- Roschke, Sebastian 138
- Sahingoz, Ozgur Koray 233
- Said, Abas Md. 243
- Salimi, Hadi 23
- Salo, Joonas 168
- Sarvanko, Jouni 158
- Sasin, Szymon 274
- Sauget, Marc 188
- Sharifi, Mohsen 23
- Shelby, Zach 274
- Shen, Yao 42
- Shichinohe, Takahiro 105
- Sonmez, Ahmet Coskun 233
- Suhonen, Jukka 254
- Taktak, Sami 127
- Tarkoma, Sasu 198
- Toninelli, Alessandra 209
- Veiga, Luís 72
- Wu, Xing 33
- Xu, Fei 52
- Yamabe, Tetsuo 105
- Ylianttila, Mika 158
- Ylioja, Jani 274
- Younis, Mohamed 243
- Yu, HeonChang 13
- Yuan, Qin 33
- Zhang, Wu 33
- Zheng, Long 42
- Zhou, Jiehan 42
- Zhou, Jingyu 42