Raja Natarajan
Adegboyega Ojo (Eds.)

LNCS 6536

Distributed Computing and Internet Technology

7th International Conference, ICDCIT 2011
Bhubaneshwar, India, February 2011
Proceedings

Springer

Raja Natarajan   Adegboyega Ojo (Eds.)

Distributed Computing and Internet Technology

7th International Conference, ICDCIT 2011
Bhubaneshwar, India, February 9-12, 2011
Proceedings

Springer

Volume Editors

Raja Natarajan
Tata Institute of Fundamental Research
School of Technology & Computer Science
Homi Bhabha Road, Colaba, Mumbai 400005, India
E-mail: raja@tifr.res.in

Adegboyega Ojo
United Nations University
International Institute of Software Technology
Center for Electronic Governance
P.O. Box 3058, Macao
E-mail: ao@iist.unu.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-19055-1 e-ISBN 978-3-642-19056-8
DOI 10.1007/978-3-642-19056-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010943036

CR Subject Classification (1998): C.2.4, C.2, D.4.2-3, D.4.7, H.2.4, H.3-5, K.6.5

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

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

Printed on acid-free paper

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

Preface

Welcome to the proceedings of the 7th International Conference on Distributed Computing and Internet Technology (ICDCIT) held during February 9–12, 2011 at the Kalinga Institute of Information Technology (KIIT) University in Bhubaneshwar, India. The conference was co-sponsored by KIIT and the Center for Electronic Governance at United Nations University—International Institute for Software Technology (UNU-IIST-EGOV), Macao. ICDCIT is an international forum for the discussion of contemporary research in distributed computing, Internet technologies, and related areas. Proceedings of all the past six ICDCIT conferences have been published in the Springer LNCS series – volume 3347 (year 2004), 3816 (2005), 4317 (2006), 4882 (2007), 5375 (2008), and 5966 (2010).

ICDCIT 2011 received 184 abstracts, of which 138 (from 12 countries) were followed by their full versions. The Programme Committee consisted of 40 members from 12 countries. Each submission was reviewed by at least two Programme Committee members, and on average by three Programme Committee members, with the help of 72 external reviewers. The Programme Committee meeting was conducted electronically over a period of two weeks in September 2010. The Programme Committee decided to accept 18 papers (13%) for presentation and publication in the LNCS proceedings. In order to make the conference more inclusive, and to provide greater scope for interesting conference presentations and discussions, the Programme Committee decided to accept 21 additional papers for presentation only. We would like to thank all the Programme Committee members for their hard work dedicated to reviews and discussions, and all the external reviewers for their invaluable contributions.

This volume also contains the full papers of six distinguished invited speakers: Jos Baeten (Eindhoven University of Technology, The Netherlands), Yves Deswarte (LAAS-CNRS, France), Kohei Honda (Queen Mary and Westfield College, UK), Vaughan Pratt (Stanford University, USA), Krishna Shankara Narayanan (IIT Bombay, India), and Maria Wimmer (University of Koblenz, Germany). We would like to thank all the invited speakers for accepting our invitations to speak and also for sending their papers.

Our thanks to Achyuta Samanta (Founder KIIT), for his support of ICDCIT 2011 and for providing the infrastructure of KIIT to organize the conference. We are grateful to KIIT and UNU-IIST for being co-sponsors of ICDCIT 2011. Our thanks to Ashok Kolaskar (Vice-Chancellor, KIIT) and Peter Haddawy, (Director, UNU-IIST) for the co-sponsorships. We are grateful to the Advisory Committee and the General Co-chairs for their invaluable support and guidance. We are indebted to Animesh Tripathy, Samaresh Mishra, Prachet Bhuyan, D.N. Dwivedy, and Hrushikesha Mohanty for their tireless efforts that made ICDCIT 2011 in Bhubaneshwar possible.

Our thanks to all the authors whose scholarly submissions offered an interesting technical program. EasyChair made the handling of submissions and the production of the proceedings extremely smooth and efficient. For the publishing process at Springer, we would like to thank Alfred Hofmann and Anna Kramer for their constant help and cooperation. We acknowledge UNU-IIST and the Tata Institute of Fundamental Research (TIFR) for providing the infrastructural support to carry out this editorial work.

Our thanks to all the participants for lively interactions that made ICDCIT 2011 enjoyable.

February 2011

Raja Natarajan
Adegboyega Ojo

Conference Organization

ICDCIT 2011 and its associated events were held at Kalinga Institute of Industrial Technology, Bhubaneshwar, India.

Sponsoring Institutions

Kalinga Institute of Industrial Technology (KIIT) University, Bhubaneshwar, India, and the Center for Electronic Governance at United Nations University—International Institute for Software Technology (UNU-IIST-EGOV), Macao.

Patron

Achyuta Samanta	KIIT, India

Advisory Committee

Maurice Herlihy	Brown University, USA
Gérard Huet	INRIA, France
Tomasz Janowski	UNU-IIST, Macao
Ashok S. Kolaskar	KIIT, India
David Peleg	WIS, Israel
R.K. Shyamasundar	TIFR, India

General Co-chairs

Hrushikesha Mohanty	University of Hyderabad, India
Vivek Sarkar	Rice University, USA

Organizing Chair

Animesh Tripathy	KIIT, India

Finance Chair

Samaresh Mishra	KIIT, India

Publicity Chair

Prachet Bhuyan	KIIT, India

Programme Chairs

Raja Natarajan	TIFR, India
Adegboyega Ojo	UNU-IIST, Macao

Programme Committee

Sowmya Arcot	UNSW, Australia
Purandar Bhaduri	IIT Guwahati, India
Nikolaj Bjorner	Microsoft, USA
Elizabeth Buchanan	University of Wisconsin-Milwaukee, USA
Antonio Cerone	UNU-IIST, Macao
Venkatesh Choppella	IIIT Hyderabad, India
Van Hung Dang	Vietnam National University, Vietnam
Elsa Estevez	UNU-IIST, Macau
Pablo Fillottrani	Universidad Nacional del Sur, Argentina
Michele G. Pinna	University of Cagliari, Italy
Veena Goswami	KIIT, India
Chittaranjan Hota	BITS Pilani, India
Paul Humphreys	University of Virginia, USA
Aditya Kanade	IISc, India
Delia Kesner	Université Paris Diderot, France
Paddy Krishnan	Bond University, Australia
Lakshmanan Kuppusamy	VIT, India
Sanjay Madria	Missouri University, USA
Rupak Majumdar	MPI, Germany
Tulika Mitra	NUS, Singapore
Debajyoti Mukhopadhyay	Calcutta Business School, India
G.B. Mund	KIIT, India
Ankur Narang	IBM, India
Rajdeep Niyogi	IIT Roorkee, India
Brajendra Panda	University of Arkansas, USA
N. Parimala	JNU, India
Manas Ranjan Patra	Berhampur University, India
Dana Petcu	West University of Timisoara, Romania
P. Radha Krishna	Infosys, India
Srini Ramaswamy	ABB Corporate Research, India
Benoit Razet	TIFR, India
Manoj Saxena	University of Delhi, India
Ashutosh Saxena	Infosys, India
Jaydip Sen	TCS, India
Manuel Serrano	INRIA, France
Hardeep Singh	GNDU, India
Hideyuki Takahashi	Tohoku University, Japan
Nobuko Yoshida	Imperial College London, UK

External Reviewers

Anirudh Santhiar	Anthony Tam	Anuraag Sridhar
Arijit Sur	Arnab De	B. Hariharan
Barbara Thoenssen	Bernhard Hengst	Carole Delporte
Christophe Prieur	D. Manjunath	Diganta Goswami
Dimitri Semenovich	Eugene Asarin	Fabien de Montgolfier
Gary Steri	Hadi Otrok	Hieu Vo
Hoang Truong	Hrushikesha Mohanty	Hugues Fauconnier
Ian Hodkinson	Indhumathi Raman	Indrajit Bhattacharya
Jonathan Hayman	Jonathan White	Jorgen Peddersen
Jyothish Soman	K. Samudravijaya	Kumar K.
Kumar Swamy H.V.	Ludovico Boratto	Madhu Viswanatham
Marcus Randall	Massimo Bartoletti	Massimo Merro
Matthieu Latapy	Mauro Piccolo	Mihaela Sighireanu
Mike Bain	Narendra Kumar Nelabhotla	Ngoc Hung Pham
Onkar Dabeer	Pawel Sobocinski	Pinaki Mitra
Pranavadatta D.N.	Qussai Yaseen	Rahul Vaze
Riccardo Scateni	S.V. Rao	Salil Kanhere
Sebastian Nanz	Souvik Bhattacherjee	Stefanie Kosuch
Sukumar Nandi	Sumita Basu	Suresh Purini
Sushanta Karmakar	T. Venkatesh	Thi Minh Chau Tran
Trong Dung Nguyen	Vamsi Krishna Brahmajosyula	Vasanta Lakshmi K.
Victor Khomenko	Vijaya Saradhi	Vikas Garg
Vivek Sarkar	Xiongcai Cai	Yacine Boufkhad
Yang Wang	Kamala Krithivasan	Zheng Da Wu

Table of Contents

Invited Talks

An Overview of Membrane Computing 1
Shankara Narayanan Krishna

Protecting Critical Infrastructures While Preserving Each Organization's Autonomy 15
Yves Deswarte

Computations and Interaction 35
Jos C.M. Baeten, Bas Luttik, and Paul van Tilburg

Scribbling Interactions with a Formal Foundation 55
Kohei Honda, Aybek Mukhamedov, Gary Brown, Tzu-Chun Chen, and Nobuko Yoshida

Open Government in Policy Development: From Collaborative Scenario Texts to Formal Policy Models 76
Maria A. Wimmer

Linear Process Algebra 92
Vaughan Pratt

Distributed Computing

Jump-Start Cloud: Efficient Deployment Framework for Large-Scale Cloud Applications 112
Xiaoxin Wu, Zhiming Shen, Ryan Wu, and Yunfeng Lin

Capacity Estimation in HPC Systems: Simulation Approach 126
A. Anghelescu, R.B. Lenin, S. Ramaswamy, and K. Yoshigoe

A Multi–Granular Lock Model for Distributed Object Oriented Databases Using Semantics 138
V. Geetha and N. Sreenath

Contention-Free Many-to-Many Communication Scheduling for High Performance Clusters 150
Satyajit Banerjee, Atish Datta Chowdhury, Koushik Sinha, and Subhas Kumar Ghosh

Recursive Competitive Equilibrium Approach for Dynamic Load Balancing a Distributed System 162
K. Shahu Chatrapati, J. Ujwala Rekha, and A. Vinaya Babu

Sensor Networks

Smoothed Functional and Quasi-Newton Algorithms for Routing in Multi-stage Queueing Network with Constraints 175
K. Lakshmanan and Shalabh Bhatnagar

An Incremental Power Greedy Heuristic for Strong Minimum Energy Topology in Wireless Sensor Networks 187
B.S. Panda and D. Pushparaj Shetty

k^{th} Order Geometric Spanners for Wireless Ad Hoc Networks 197
Prabhat Kiran and S.V. Rao

Robust and Distributed Range-Free Localization Using Anchor Nodes with Varying Communication Range for Three Dimensional Wireless Sensor Networks 209
Manas Kumar Mishra and M.M. Gore

Internet Technologies and Applications

Decision Support Web Service 221
N. Parimala and Anu Saini

A Scalable Architecture for Real-Time Online Data Access 232
Ionuţ Roşoiu

Socially Responsive Resource Usage: A Protocol 243
Hrushikesha Mohanty

An Automated HSV Based Text Tracking System from Complex Color Video 255
C. Misra and P.K. Swain

Security

Enhanced Insider Threat Detection Model that Increases Data Availability 267
Qussai Yaseen and Brajendra Panda

Checking Anonymity Levels for Anonymized Data 278
V. Valli Kumari, N. Sandeep Varma, A. Sri Krishna, K.V. Ramana, and K.V.S.V.N. Raju

Chaos Based Image Encryption Scheme Based on Enhanced Logistic Map 290
I. Shatheesh Sam, P. Devaraj, and R.S. Bhuvaneswaran

Bio-inspired Computing

Matrix Insertion-Deletion Systems for Bio-Molecular Structures 301
Lakshmanan Kuppusamy, Anand Mahendran, and Shankara Narayanan Krishna

Artificial Bee Colony Based Sensor Deployment Algorithm for Target Coverage Problem in 3-D Terrain 313
S. Mini, Siba K. Udgata, and Samrat L. Sabat

Author Index 325

An Overview of Membrane Computing

Shankara Narayanan Krishna

Department of Computer Science and Engineering,
IIT Bombay, Powai, Mumbai, India 400 076
krishnas@cse.iitb.ac.in

Abstract. Membrane Computing is a *natural computing* paradigm aiming to abstract computing models from the structure and functioning of the living cell as well as from the cooperation of cells in tissues, organs and other populations of cells. This direction of research was initiated by Gh. Păun in November 1998 [25]. In the last twelve years, the area has grown substantially: initial research focussed on understanding computability aspects using formal language theoretic elements, and using membrane computing as a parallel computing device capable of solving intractable problems; over the years, membrane computing has been found useful in modelling biological processes, simulating ecosystems, and also finds some applications in areas like economics, computer graphics and approximate optimization. Off late, complexity classes (time, space) of membrane systems and their connection with the classical complexity classes have been investigated. The connection of membrane computing with other areas like petri nets, brane calculi, process algebra, dynamical systems, X-machines and models based on fuzzy sets is an active and important recent line of research. In this paper, we give a high level overview of the research in membrane computing over the last 12 years.

1 Introduction

The research area of membrane computing originated as an attempt to formulate a model of computation motivated by the structure and functioning of the living cell - more specifically, by the role of membranes in partitioning living cells into individual "reaction agents". The initial model was based on a hierarchical arrangement of *membranes* delimiting compartments, where *multisets* of objects (representing proteins, molecules and chemicals) evolve according to given *evolution rules*. These rules were either capturing chemical reactions and had the form of multiset rewriting rules, or were inspired by other biological mechanisms, like communication of objects through membranes (a classic example is the symport/antiport action in systems) and had the form of communication rules. The initial model was modified later to incorporate additional features motivated by biological/mathematical/computer science considerations. One such modification was that the hierarchical model was replaced by a non-hierarchical arrangement of membranes. While hierarchical (cell-like) arrangements of membranes correspond to trees, the non-hierarchical (tissue-like) arrangements correspond to arbitrary graphs as underlying structures, with membranes placed on the nodes and edges corresponding to communication channels. The most recent developments in this line are the neural-like membrane systems motivated by spiking neural networks.

All the computing devices considered in membrane computing are called *P Systems* in honour of Gh.Păun, the father of the area. In the initial phases of research, investigation of computability aspects of P systems led to several variants of P systems turning out to be equivalent to Turing machines, hence computationally complete. This was followed by research concerning complexity classes yielding results relating classic complexity classes to computational efficiency of various variants of P systems. Research on applications of P systems is currently a very active area, with applications in several areas, to mention in particular biology and bio-medicine.

The paper is organized as follows: In section 2, we introduce [25] the initial hierarchical cell-like model, illustrate the working of a P system using an example, and discuss computability results. We close section 2 mentioning some well known variants of P systems. In section 3, we introduce a variant of P systems modeling the biological action *mitosis*. This is the first known variant of P systems capable of solving intractable problems. We talk about complexity classes for P systems, and do a quick survey of complexity related results. We conclude in section 4 mentioning some applications.

2 The Basic Model

In this section, we recall the definition of the basic model of P systems as introduced in [25]. A *membrane* is a three dimensional vesicle, which geometrically can be considered as a ball in the Euclidean space. A membrane thus delimits space, separating the inside from the outside. The inside space serves as a reactor - where reactions take place using the molecules/chemicals in the space. In the basic model, we will consider a hierarchical arrangement of nested membranes. Graphically, we represent membrane structures as Euler-Venn diagrams. Figure 1 gives a membrane structure.

In the figure, the membranes are labeled in a 1-1 manner through numbers ranging from 1 to 10. The outermost membrane labeled 1 is called the *skin* membrane. The space delimited by membrane 1 and membranes 2,3 and 6 is called the *region* of membrane 1. Likewise, the space delimited by membrane 2 is the region of membrane 2 and so on. The space outside the skin membrane is called the *environment*. The membranes 2, 3 and 6 are the children of membrane 1, membrane 2 has no children, membrane 3 has children 4 and 5 and so on. A membrane is called *elementary* if it has no children. The skin membrane is unique. A region is either the space delimited by an elementary membrane or a space delimited by a non-elementary membrane and its children. The membranes 4,5 are siblings; so are membranes 8,9 and 10. It is easy to visualize a membrane structure as a rooted tree, with the root corresponding to the skin membrane. An edge from node i to node j corresponds to membrane j being a child of

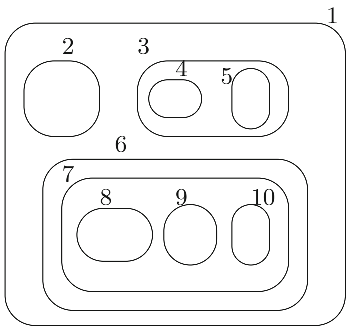


Fig. 1. A membrane structure

membrane i. The leaf nodes are the elementary membranes. Commonly, to save space, a linear parentheses expression is adopted to specify membrane structures. Thus, the parenthesis expression corresponding to the membrane structure in Figure 1 is

$$[_1\ [_2\ ]_2\ [_3\ [_4\ ]_4\ [_5\ ]_5\ ]_3\ [_6\ [_7\ [_8\ ]_8\ [_9\ ]_9\ [_{10}\ ]_{10}\ ]_7\ ]_6\ ]_1$$

Next, we come to multisets of objects which can be placed in the regions delimited by the membranes and rules to process them. In an abstract form, we assume that the proteins/molecules present in regions are represented by multisets over a finite alphabet Σ. The multisets of objects in region i is represented by w_i. The rules for processing these objects are of the form $u \to v$ where u is a string and v is a string over $\Sigma \times tar$, $tar \in \{here, in, out\}$ represents the target membrane. For example, in the above case, we can have $\Sigma = \{a, b, c, d\}$ with $w_1 = aa, w_2 = bc, w_3 = a = w_4, w_5 = c, w_6 = \emptyset = w_7, w_8 = ab, w_9 = bc, w_{10} = cc$. We can represent this as part of the membrane structure as

$$[_1 aa\ [_2\ bc\ ]_2\ [_3\ a\ [_4\ a\ ]_4\ [_5\ c\ ]_5\ ]_3\ [_6\ [_7\ [_8\ ab\ ]_8\ [_9\ bc\ ]_9\ [_{10}\ cc\ ]_{10}\ ]_7\ ]_6\ ]_1$$

Rules processing objects w_i are denoted by R_i. We can have in this example, $R_1 = \{a \to (b, out)\}$, $R_2 = \{b \to (c, here), c \to (a, here)\}$, $R_3 = \{a \to (cc, in)\}, R_4 = \{(a \to (b, out)\}$, $R_5 = \{c \to (a, here)\}$, $R_6 = \{b \to (a, here)\}$, $R_7 = \emptyset, R_8 = \{a \to (b, here), b \to (c, here)\}$, $R_9 = \emptyset, R_{10} = \{c \to (a, out)\}$. The rules are applied in the *maximally parallel* mode; i.e, all objects to which a rule is applicable are processed in a step, if more than one rule is applicable to an object, then one rule is chosen non-deterministically and applied. A configuration of the system consists of the membrane structure and their contents at a given point of time. The initial configuration C_0 is given by the membrane structure and the multisets of all regions. A configuration C_i evolves to a configuration C_{i+1} by one rewriting step, used in the maximally parallel mode. The string above represents the initial configuration C_0. It is possible to have multiple configurations corresponding to a rewriting step depending on the choice of the rules used. In our example, C_1 is the configuration

$$bb[_1\ [_2\ ca\ ]_2\ [_3\ b\ [_4\ c\ ]_4\ [_5\ ca\ ]_5\ ]_3\ [_6\ [_7\ aa\ [_8\ bc\ ]_8\ [_9\ bc\ ]_9\ [_{10}\ ]_{10}\ ]_7\ ]_6\ ]_1$$

Note that while using the rule $a \rightarrow (cc, in)$ in R_3, we have non-deterministically distributed the two c's among membranes 4,5 (each get a copy of c). From C_1, we obtain the next configuration as C_2 which is

$$bb[_1\ [_2\ aa\ ]_2\ [_3\ b\ [_4\ c\ ]_4\ [_5\ aa\ ]_5\ ]_3\ [_6\ [_7\ aa\ [_8\ cc\ ]_8\ [_9\ bc\ ]_9\ [_{10}\ ]_{10}\ ]_7\ ]_6\ ]_1$$

From C_2, no more evolution is possible, and the system *halts*. To observe the *output* of a halting computation, we designate a membrane or the environment and observe the contents of that membrane (or environment) after halting. If we designate membrane 10 as the output membrane, then at the end of a halting computation, we obtain $\emptyset$ as the result. If we consider the environment, then we obtain bb. Formally, a P system with multiset rewriting rules of degree $m \geq 1$ is a construct

$$\Pi = (\Sigma, H, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0)$$

where

1. Σ is the finite alphabet of objects,
2. H is the alphabet of membrane labels,
3. μ is a membrane structure with m membranes (degree m),
4. $w_1, \ldots, w_m \in \Sigma^*$ are the multisets of objects associated with the m regions of μ,
5. R_i, $1 \leq i \leq m$ are finite sets of multiset rewriting rules associated with the m regions of μ,
6. $i_0 \in H \cup \{env\}$ specifies the output region of Π, where env stands for the environment.

The number or the Parikh vector of the multiset of objects from Σ contained in i_0 at the moment when the system halts is the *result* of Π. By collecting the results of all possible computations possible in Π we get the set of natural numbers and vectors generated by Π, denoted by $\mathbf{N}(\Pi)$ and $Ps(\Pi)$ respectively. In the above example, when we consider the environment as i_0, we obtain $Ps(\Pi) = \{(0, 2, 0)\}$ and $\mathbf{N}(\Pi) = 2$. The families of all sets of numbers or sets of vectors computed by P systems with multiset rewriting with atmost m membranes is denoted by $\mathbf{N}OP_m$ and $PsOP_m$ respectively. By allowing only "minimal" cooperation among the objects, these systems have been shown to be computationally complete. Formally, using rules of the form $ca \rightarrow cv$ and $a \rightarrow v$, where c is a specified fixed object outside Σ, $a \in \Sigma$, and v a string over $\Sigma \times \{here, out, in\}$, we obtain with just two membranes the result

Theorem 1. *$\mathbf{N}OP_2 = \mathbf{N}RE$.*

where $\mathbf{N}RE$ denotes the family of Turing computable sets of numbers. The proof can be seen in [25], [34].

2.1 Variants of the Basic Model

Following the initial model of P systems with multiset objects, several variants were introduced. All the results mentioned here can be found in [34]. We mention some of the prominent ones here:

1. P systems with string objects and rewriting rules [25]. The computing power of the model is established by comparing with families RE, CF, MAT and ET0L. Using various features like permitting/forbidding contexts, membrane dissolution, membrane permeability, the computing power is investigated. Certain combinations of these features provide computational completeness. The application of rules in sequential/parallel mode and their effect on the computational power has been studied. Decision questions such as reachability of a deadlock configuration has also been investigated. An important extension of P systems with rewriting rules is the one where replication is introduced [17]. This variant is capable of producing an exponential workspace in polynomial time, and is thus useful for solving intractable problems.
2. Splicing P systems [25]. This variant is motivated by the splicing operation. The biological splicing operation was first mathematically formalized by T. Head [11]. Using the rotate and simulate technique, a characterization of RE is obtained using 2 membranes. By a transformation of splicing P systems into language equivalent H systems [24], results comparing the power of splicing P systems with the Chomsky hierarchy are obtained. Restricted variants of splicing P systems have been studied by considering features like one-way communication and immediate communication [34]. The construction of a universal splicing P system can be seen in [10].
3. P systems with communication rules [23]. This variant has multisets of objects, but no rewriting rules of any kind. The only rules governing the system are the communication of objects across membranes inspired by the biological operations *symport* and *antiport*. Various modes of parallelism in the use of rules and various modes of halting are considered, and Turing completeness is achieved with some combinations. The size of the symport/antiport rules (number of objects per communication) is a parameter which also influences the computational power. A trade off between the different parameters, viz., number of membranes and number of objects per communication is studied, and its effect on the computing power is analyzed. There are open questions regarding the descriptional complexity measures for this class of P systems based on the parameters [34].
4. Tissue and population P systems [3], [9], [19]. Tissue P systems were introduced as a generalization of cell-like P systems [25] by considering a more general structure than a tree, and population P systems are an extension of these. Tissue P systems have a generic graph as the underlying structure and pure communication governs the evolution of the system. Specific channels of communication are established between various membranes and between the membranes and the environment. The computational efficiency is analyzed using the complexity principles of number of symbols, number of cells, size of rules involved and different modes of parallelism. While tissue P systems have a fixed underlying cell topology, in population P systems, the structure can dynamically change. The quorum sensing behaviour of *Vibrio fischeri* bacterium has been successfully modeled using population P systems.

5. P systems with active membranes [26]. This variant of P systems model the biological action of *mitosis* in cells. This was the first variant to solve intractable problems in polynomial time apart from being Turing complete. Several restrictions of the model introduced in [26] have been studied with respect to computational efficiency, solving hard problems, and time, space complexity considerations. Related variants are [2] with dynamic creation of membranes modeling the biological process of *autopoiesis* and [14] with mobile membranes.
6. Spiking neural P systems [12]. This variant is inspired by the way neurons communicate by means of electric impulses of identical shape, called *spikes*. The underlying structure of such a system is a directed graph with neurons placed on the nodes. The synapses between the neurons determine the connections of the graph. Closure properties, comparisons with the Chomsky hierarchy as well as the ability to solve intractable problems are the highlights of this variant.
7. P systems with objects on membranes [6], [22]. This model is a deviation from the standard model where objects are placed in the membranes. [22] models *peripheral* proteins placed on sides of the membranes or *integral* proteins which have parts of the molecule on both sides of the membrane. In this framework, the area of membrane computing has been compared [15] to that of brane calculi [8]. Decision questions regarding reachability of configurations in this model has been investigated in [7].

In general, computational power has been investigated by simulation of traditional devices (register machines, partially blind counter machines, vector addition systems, suitable grammars). The first paper to differ from this and prove computational completeness is [16]. In [16], P systems with mobile membranes simulate with a linear slow down (without introducing non-determinism) P systems with replicated rewriting, thereby achieving Turing completeness. The fact that a variant with symbol objects simulates a variant with string objects is a hallmark of this work. The question of biological feasibility of such simulations for various variants is an interesting question.

3 Complexity Aspects

While the previous section discussed about the computational power of various variants of P systems, this section addresses how real life problems can be solved by P systems. To this aim, notions of classical computational complexity theory are adapted for the membrane computing framework. Complexity aspects of basic transition P systems [25], P systems with active membranes [26], P systems with membrane creation [2] and spiking neural P systems [12] have been investigated so far. The efficiency of these systems and their relations with classes **P**, **NP**, **NP** $\cap$ **co-NP**, **L**, **NL** and **PSPACE** have been investigated. In this section, we look only at the results concerning P systems with active membranes [26]. We first recall the definition.

3.1 Active Membranes

A P system with active membranes of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, H, \mu, w_1, \ldots, w_q, R, h_0)$ where $\Gamma, H, w_1, \ldots, w_q, h_0$ are as in the basic model in section 2, μ is a membrane structure consisting of q membranes injectively labeled with elements of H and with electrical charges $(+, -, 0)$ associated with them, and R is a finite set of rules, of the following forms:

1. $[_h a \to u]^\alpha_h$ for $h \in H$, $\alpha \in \{+, -, 0\}$, $a \in \Gamma, u \in \Gamma^*$ (object evolution rules),
2. $a[_h]^\alpha_h \to [_h b]^\beta_h$, for $h \in H$, $\alpha, \beta \in \{+, -, 0\}$, $a, b \in \Gamma$ (send-in communication rules),
3. $[_h a]^\alpha_h \to [_h]^\beta_h b$, for $h \in H$, $\alpha, \beta \in \{+, -, 0\}$, $a, b \in \Gamma$ (send-out communication rules),
4. $[_h a]^\alpha_h \to b$, for $h \in H$, $\alpha \in \{+, -, 0\}$, $a, b \in \Gamma$ (dissolution rules),
5. $[_h a]^\alpha_h \to [_h b]^\beta_h [_h c]^\zeta_h$, for $h \in H$, $\alpha, \beta, \zeta \in \{+, -, 0\}$, $a, b, c \in \Gamma$ (division rules for elementary membranes),
6. $[_h [_{h_1}]^\alpha_{h_1} \cdots [_{h_k}]^\alpha_{h_k} [_{h_{k+1}}]^\beta_{h_{k+1}} \cdots [_{h_n}]^\beta_{h_n}]^\theta_h \to [_h [_{h_1}]^\gamma_{h_1} \cdots [_{h_k}]^\gamma_{h_k}]^\zeta_h [_h [_{h_{k+1}}]^\kappa_{h_{k+1}} \cdots [_{h_n}]^\kappa_{h_n}]^\chi_h$, for $h, h_1, \ldots, h_n \in H$, $\alpha, \beta, \theta, \gamma, \zeta, \kappa, \chi \in \{+, -, 0\}$, and $\{\alpha, \beta\} = \{+, -\}$ (division rules for non-elementary membranes).

When we treat membrane systems as language deciding devices to solve decision problems, we call these *recognizer membrane systems*. Each computation of a recognizer membrane system outputs either an object *yes* or an object *no*. The output is read only at the halting configuration. Formally, a *recognizer membrane system* Π is such that (1) All computations halt, (2) $yes, no \in \Gamma$, and (3) One of the objects yes, no appear in the halting configuration. The computation is accepting if the *yes* object arrives in the halting configuration and rejecting if the *no* object arrives. Similar to circuit complexity, we consider an infinite family of active membrane P systems Π to solve a decision problem. Let $X = \{x_1, x_2, \ldots\}$ be a language over alphabet Σ. The family Π decides X if for any string $x \in \Sigma^*$, the P system $\Pi(x)$ accepts whenever $x \in X$ and rejects otherwise. Thus, each instance of the problem is solved by some family member of Π. The family Π is *sound* with respect to X when, for each $x \in \Sigma^*$, if there exists an accepting computation of $\Pi(x)$, then $x \in X$. The family Π is *complete* with respect to X when, for each $x \in \Sigma^*$, if $x \in X$, then every computation of $\Pi(x)$ is accepting. A membrane system is *confluent* if it is sound and complete.

3.2 Uniformity and Semi-uniformity

The notion of *uniformity* was first introduced by Borodin [5] for boolean circuits. In this section, we discuss notions of uniformity and semi-uniformity applied to membrane systems. With no restrictions, recognizer active membrane systems are a non-uniform model of computation; that is, there may be a different device solving the problem for each input size. This idea is similar to the case of employing different circuits to solve different problem instances; we consider an infinite family of recognizer active membrane systems to cover all potential input strings. However, if we can invest unbounded amounts of computation in order to construct each member of the family, it can potentially solve uncomputable problems. To ensure that the function that constructs each member of the family does not increase the set of problems decided by the family, we impose that the constructing function is computable within certain restricted resources (time/space). When the function maps a single input length to a membrane system that decides all inputs of that length, then the function is called a *uniformity condition*. When the function maps a single input word to a membrane system that decides that input, then the function is called a *semi-uniformity* condition. The notions of uniformity and semi-uniformity were first applied to membrane systems in [27].

Class of problems solved by a uniform family: Let $\mathcal{R}$ be a family of recognizer membrane systems and let $t : \mathbf{N} \to \mathbf{N}$ be a total function. Let E, F be classes of functions. The class of problems solved by an (E, F)-uniform family of membrane systems $\mathcal{R}$ in time t denoted $(E, F) - MC_{\mathcal{R}}(t)$ contains all problems X such that

(a) There exists a F-uniform family of membrane systems $\mathbf{\Pi} = \{\Pi_1, \Pi_2, \ldots\}$ of type $\mathcal{R}$: that is, there exists a function $f \in F$, $f : \{1\}^* \to \mathbf{\Pi}$ such that $f(1^n) = \Pi_n$,
(b) There exists an input encoding function $e \in E$ such that $e(x)$ is the input multiset of Π_n, for $|x| = n$,
(c) $\mathbf{\Pi}$ is t-efficient: Π_n halts in $t(n)$ steps,
(d) The family $\mathbf{\Pi}$ is sound with respect to (X, e, f): for each $x \in X$, there is an accepting computation of $\Pi_{|x|}$ on input $e(x)$,
(e) The family $\mathbf{\Pi}$ is complete with respect to (X, e, f): for each $x \in X$, every computation of $\Pi_{|x|}$ on input $e(x)$ must be accepting.

The set of languages decided by a uniform family of membrane systems in polynomial time is defined as

$$(E, F) - PMC_{\mathcal{R}} = \bigcup_{k \in \mathbf{N}} (E, F) - MC_{\mathcal{R}}(n^k)$$

Semi-uniformity is a generalization of uniformity. Let H be a class of functions. The class of problems solved by a (H)-semi-uniform family of membrane systems of type $\mathcal{R}$ in time t denoted $(H) - MC^*_{\mathcal{R}}(t)$ contains all problems X such that condition (c) as above as well as soundness and completeness hold, and in place of condition (a) we have the following: there exists a H-semi-uniform family $\mathbf{\Pi} = \{\Pi_{x_1}, \Pi_{x_2}, \ldots\}$ of type $\mathcal{R}$: there exists $h \in H$, $h : \Sigma^* \to \mathbf{\Pi}$ such that $h(x_i) = \Pi_{x_i}$. The set of languages decided by a semi-uniform family of membrane systems in polynomial time is defined as $(H) - PMC^*_{\mathcal{R}} = \bigcup_{k \in \mathbf{N}} (H) - MC^*_{\mathcal{R}}(n^k)$. In our case, $\mathcal{R}$ will be substituted with various (sub)classes of P systems with active membranes. Let $\mathcal{NAM}$ denote the class of P systems with active membranes where division rules are not used. The class $\mathbf{P}$ characterizes both uniform and semi-uniform families of recognizer P systems with active membranes without division rules, when the encoding as well as construction functions $\in \mathbf{P}$.

Theorem 2. *[34]* $(P)PMC^*_{\mathcal{NAM}} = (P,P)PMC_{\mathcal{NAM}} = \mathbf{P}$.

Let $\mathcal{AM}(+n)$ (respectively $\mathcal{AM}(-n)$) denote the class of recognizer P systems with active membranes allowing division for both elementary and non-elementary membranes (respectively division only for elementary membranes). In the framework of $\mathcal{AM}(-n)$, efficient uniform solutions [34] to weakly **NP**-complete problems (knapsack, subset sum, partition) and strongly **NP**-complete problems (SAT, clique, bin packing) have been obtained.

A linear time solution to SAT by a uniform family of recognizer P systems with active membranes $\mathcal{AM}(-n)$ is given in [34]. This result, along with the observation that $(P,P)PMC_{\mathcal{R}}$ is closed under complements and polynomial time reductions gives us the following result:

Theorem 3. *1.* $SAT \in (P,P)PMC_{\mathcal{AM}(-n)}$,
2. $\mathbf{NP} \cup \mathbf{co-NP} \subseteq (P,P)PMC_{\mathcal{AM}(-n)}$.

The class of P systems with active membranes where non-elementary division of membranes is allowed, has been shown to solve a **PSPACE** complete problem, QBF. The upper and lower bounds of this class [1], [29], [32] are

Theorem 4. $\mathbf{PSPACE} \subseteq (P,P)PMC_{\mathcal{AM}(+n)} \subseteq (P)PMC^*_{\mathcal{AM}(+n)} \subseteq \mathbf{EXP}$.

3.3 Avoiding Polarizations

Next, several classes of recognizer P systems without electrical charges and with different kinds of membrane division rules were studied from a computational complexity point of view. All rules in section 3.1 carry over without polarizations. In particular, we mention rules (5) and (6) in this context:
(5)$[_h a]_h \rightarrow [_h b]_h [_h c]_h$, for $a,b,c \in \Gamma, h \in H$,
(weak division rules for elementary/non-elementary membranes)
(6)$[_h [_{h_1}]_{h_1} \cdots [_{h_k}]_{h_k} [_{h_{k+1}}]_{h_{k+1}} \cdots [_{h_n}]_{h_n}]_h \rightarrow [_h [_{h_1}]_{h_1} \cdots [_{h_k}]_{h_k}]_h [_h [_{h_{k+1}}]_{h_{k+1}} \cdots [_{h_n}]_{h_n}]_h$, for $h, h_1, \ldots, h_n \in H$, $k \geq 1, n \geq k$
(strong division rules for non-elementary membranes).
Throughout, we use rule (6) only for the case $k = 1, n = 2$. A restriction of rule (6) is when the division happens in the presence of a specific membrane labeled p that is, rules of the form
(7) $[_h [_{h_1}]_{h_1} [_{h_2}]_{h_2} [_p]_p]_h \rightarrow [_h [_{h_1}]_{h_1} [_p]_p]_h [_h [_{h_2}]_{h_2} [_p]_p]_h$.
The class of recognizer polarizationless P systems with active membranes (resp. without division) is denoted $\mathcal{AM}^0$ (resp. $\mathcal{NAM}^0$) and $\mathcal{AM}^0(\alpha, \beta, \gamma, \delta)$ where the parameters are as follows:

(a) $\alpha \in \{+d, -d\}$. $+d(-d)$ stand for allowing (disallowing) dissolution rules,
(b) $\beta \in D = \{-n, +nw, +ns, +nsw, +nsr\}$. $-n$ stands for disallowing non-elementary division, $+nw(+ns)$ for weak (strong) division for elementary and non-elementary membranes, $+nsw$ for weak and strong division for elementary/non-elementary membranes and $+nsr$ for allowing rules of type (5), (6) and (7),
(c) $\gamma \in \{+e, -e\}$. $+e(-e)$ stand for allowing (disallowing) evolution rules, and
(d) $\delta \in \{+c, -c\}$. $+c(-c)$ stand for allowing (disallowing) communication rules.

The following results can be found in [34].

Theorem 5. *1.* $\mathbf{P} \subseteq (P,P)PMC_{\mathcal{NAM}^0(-d,-e,+c)}$,
2. $\mathbf{P} = (P)PMC^*_{\mathcal{AM}^0(-d,\beta,-e,+c)}$, *with* $\beta \in D$,
3. $\mathbf{NP} \cup \mathbf{co-NP} \subseteq (P)PMC^*_{\mathcal{AM}^0(+d,+ns,+e,+c)}$,
4. $\mathbf{NP} \cup \mathbf{co-NP} \subseteq (P)PMC^*_{\mathcal{AM}^0(+d,+nsw,-e,-c)}$,
5. $\mathbf{PSPACE} \subseteq (P,P)PMC_{\mathcal{AM}^0(+d,+ns,+e,+c)}$,
6. $\mathbf{PSPACE} \subseteq (P)PMC^*_{\mathcal{AM}^0(+d,+nsr,-e,-c)}$.

So far, in all the results connecting complexity classes with the set of problems solvable by (semi) uniform families of P systems, one considered the class of functions E, F (or H) to be in $\mathbf{P}$. It was shown in [20] that, by tightening the uniformity conditions, it is possible to obtain characterizations of P systems by families lower than $\mathbf{P}$. In particular, [20] showed that by considering C-uniformity where $C \in \{\mathbf{AC^0}, \mathbf{NC^1}, \mathbf{L}, \mathbf{NL}\}$, polarizationless recognizer P systems with active membranes characterize the class $\mathbf{NL}$.

Theorem 6. *[20] Let* $C \in \{\mathbf{AC^0}, \mathbf{NC^1}, \mathbf{L}, \mathbf{NL}\}$. *Then for all* $\beta \in D$, $(C)PMC^*_{\mathcal{AM}^0(-d,\beta,+e,+c)} = \mathbf{NL}$.

An important question relating to complexity classes in membrane computing is with respect to the notions of semi-uniformity and uniformity. It has been observed that in almost all cases where a semi-uniform solution for a problem was given, at a later point a uniform solution was also proposed [20]. The question of interest is whether the notions of semi-uniformity and uniformity coincide in membrane systems. [20] has answered this in negative, by giving a semi-uniform class of recognizer P systems that solves a strictly larger class of problems than the corresponding uniform class. This result is summarized as

Theorem 7. *Let* $\beta \in D$. *Then* $\mathbf{AC}^0 = (\mathbf{AC}^0, \mathbf{AC}^0)PMC_{\mathcal{AM}^0(-d,\beta,+e,+c)} \subset (\mathbf{AC}^0)PMC^*_{\mathcal{AM}^0(-d,\beta,+e,+c)} = \mathbf{NL}$.

The relationship between complexity classes and the efficiency of several variants of P systems is an active line of recent research. Most of the existing work has been concerning time complexity classes, characterizations dealing with space complexity classes need to be worked out as well.

4 Discussion

In this section, we talk about recent developments in membrane computing as well as the connection between membrane computing and related areas.

4.1 Extensions, Applications

1. Dynamic probabilistic P systems (DPP) is an extension of P systems considered in [28]. This model has been found useful in modeling biological systems.

The gene regulation system of the *lac operon* in *E.coli* has been modeled using this variant [30]. This variant has also been employed to study signalling cascade pathways [21], and to provide an artificial life system that behaves very close to a quorum sensing system as exhibited by *Vibrio fischeri* [31]. A much more complex example [4] deals with the Ras/cAMP/PKA pathway in Yeast S.*cerevisiae*. This pathway is involved in the control of the yeast cell metabolism, stress resistance, and proliferation, in relation to the quantity of available nutrients.

2. Metabolic P systems (MP) are an extension proposed in [18] which introduce a new possibility in modeling complex phenomena which is related to the *log-grain* theory. In these systems, the dynamics is computed by suitable recurrent equations, based on flux regulation maps, and the log-grain theory provides a method for deducing adequate flux maps of an MP model, by means of suitable algebraic manipulations of data coming from macroscopic observation of the system to be modeled. A strong connection can be stated between MP systems and ordinary differential equations: from a differential model, an equivalent MP system can be deduced, and conversely, any MP system can be transformed into an equivalent differential model too.
3. [33] is a collection of articles dealing with applications of P systems. We mention a few here. (i) The activity of mechanosensitive channels of large conductance in cellular membranes has been modeled in P systems and an implementation *in silico* carried out, (ii) The T cell signalling network which plays a central role in cell-mediated immunity has been modeled using P systems, and an implementation of T cell signalling networks has also been carried out using this modelling. The experiments conducted gave relevant biological information on T cell behaviour, particularly T cell responses. The simulations explain how various factors play a role in determining T cell response, relating input and output values of T cell mechanisms. (iii) A model of light reactions taking place in photo synthesis is constructed using P systems. Behaviours of the model under various parameters are tested on a computer. Computer simulations show that the model explains in a good way many phenomena of photosynthesis, including photoinhibition mechanisms. A dynamical system using differential equations for photosynthesis is compared with the P system model. The comparison shows that P systems are better tools for dealing with biological phenomena than models based on differential equations using P systems.

4.2 Comparison with Related Areas

1. The relationship between membrane computing and Brane calculi has been explored in [34]. Brane calculi was introduced in [8] as a process calculi with dynamic nested membranes. Many of the basic operations used in membranes like *endo, exo, phago, pino, bud, mate, drip, wrap* were represented in the process calculi framework, their structural congruences as well as decidability of reachability, universal and existential termination under (i) maximal parallel and (ii) interleaved, sequential semantics were studied. The expressiveness

and decidability results of both the areas membrane computing and brane calculi are compared.

2. The relationship between Petri nets and membrane computing has been explored [34]. Petri nets are an operational model for concurrent systems directly generalizing state machines by their distributed states and local actions. A translation between basic membrane systems and PT nets, a prominent petri net model has been given in [13]. As a consequence, tools and techniques developed for petri nets become available for the description, analysis and behavioural verification of P systems.

4.3 Implementation Efforts

Software simulators for automatically simulating and verifying behaviours of membrane systems have been created in the past decade. [34] gives a detailed account of first and second generation software simulators. We mention two most significant directions here. P-lingua [35] is a programming language created with the aim of becoming the standard representation and implementation of future software. Programs in P-lingua define families of P systems in a parametric and modular way. The compilation tool generates an XML document associated to the P system, which can be integrated into other applications. The XML specification of a P system can be translated into an executable representation. The second big project that could have significant impact is being carried out in the Chemical faculty of Technion Institute, Haifa, Israel. It will be the first *in vitro* experiment, using test tubes as membranes and DNA molecules as objects, evolving under the control of enzymes. The group working at Technion has a big dream : that of constructing a *P Computer* which will be one of the fastest super computers; this dream is justified due to the theoretical results obtained for P systems where several variants have been shown to be RE, as well as due to the fact that hard problems can be efficiently solved by P systems. Of course, there are a lot of challenges here, the most basic being to determine and decide a feasible P system model for implementation.

References

1. Alhazov, A., Martin-Vide, C., Pan, L.: Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes. Fundamenta Informaticae 58, 67–77 (2003)
2. Arroyo, F., Baranda, A., Castellanos, J., Păun, G.: Membrane Computing: The power of (rule) creation. Journal of Universal Computer Science 8, 369–381 (2002)
3. Bernardini, F., Gheorghe, M.: Population P Systems. Journal of Universal Computer Science 10, 509–539 (2004)
4. Besozzi, D., Cazzaniga, P., Pescini, D., Mauri, G., Colombo, S., Martegani, E.: Modeling and stochastic simulation of the Ras/cAMP/PKA pathway in the yeast sacharomyces cerevisiae evidences a key regulatory function for intracellular guanine nucleotides pools. Journal of BioTechnology 133, 377–385 (2008)
5. Borodin, A.: On relating time and space to size and depth. SIAM Journal of Computing 6(4), 733–744 (1977)

6. Brijder, R., Cavaliere, M., Riscos-Núñez, A., Rozenberg, G., Sburlan, D.: Membrane systems with marked membranes. Electronic Notes in Theoretical Computer Science 171, 25–36 (2007)
7. Brijder, R., Cavaliere, M., Riscos-Núñez, A., Rozenberg, G., Sburlan, D.: Membrane systems with proteins embedded in membranes. Theoretical Computer Science 404, 26–39 (2008)
8. Cardelli, L.: Brane calculi. In: Danos, V., Schachter, V. (eds.) CMSB 2004. LNCS (LNBI), vol. 3082, pp. 257–278. Springer, Heidelberg (2005)
9. Freund, R., Păun, G., Pérez-Jiménez, M.J.: Tissue-like P systems with channel states. Theoretical Computer Science 330, 101–116 (2005)
10. Frisco, P., Hoogeboom, H.J., Sant, P.: A direct construction of a universal P system. Fundamenta Informaticae 49, 103–122 (2002)
11. Head, T.: Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. Bulletin of Mathematical Biology 49, 737–759 (1987)
12. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae 71, 279–308 (2006)
13. Kleijn, J., Koutny, M., Rozenberg, G.: Towards a petri net semantics for membrane systems. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 292–309. Springer, Heidelberg (2006)
14. Krishna, S.N.: The Power of Mobility: Four Membranes Suffice. In: Cooper, S.B., Löwe, B., Torenvliet, L. (eds.) CiE 2005. LNCS, vol. 3526, pp. 242–251. Springer, Heidelberg (2005)
15. Krishna, S.N.: Membrane computing with transport and embedded proteins. Theoretical Computer Science 410(4-5), 355–375 (2009)
16. Krishna, S.N., Păun, G.: P systems with mobile membranes. Natural Computing 4, 255–274 (2005)
17. Krishna, S.N., Rama, R.: P systems with replicated rewriting. Journal of Automata, Languages and Combinatorics 6, 345–350 (2001)
18. Manca, V., Bianco, L., Fontana, F.: Evolutions and oscillations of P systems: Applications to biological phenomena. In: Mauri, G., Păun, G., Jesús Pérez-Jímenez, M., Rozenberg, G., Salomaa, A. (eds.) WMC 2004. LNCS, vol. 3365, pp. 63–84. Springer, Heidelberg (2005)
19. Martin-Vide, C., Păun, G., Pazos, J., Rodriguez-Paton, A.: Tissue P Systems. Theoretical Computer Science 296, 295–326 (2003)
20. Murphy, N.: Uniformity conditions for membrane systems: Uncovering complexity below **P**. Ph.D thesis, National University of Ireland, Maynooth (May 2010)
21. Păun, A., Pérez-Jiménez, M.J., Romero-Campero, F.J.: Modelling signal transduction using P systems. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2006. LNCS, vol. 4361, pp. 100–122. Springer, Heidelberg (2006)
22. Păun, A., Popa, B.: P systems with proteins on membranes. Fundamenta Informaticae 72, 467–483 (2006)
23. Păun, A., Păun, G.: The power of communication: P systems with symport/antiport. New Generation Computing 20, 295–305 (2002)
24. Păun, G.: DNA Computing: distributed splicing systems. In: Mycielski, J., Rozenberg, G., Salomaa, A. (eds.) Structures in Logic and Computer Science. LNCS, vol. 1261, pp. 353–370. Springer, Heidelberg (1997)
25. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000); First circulated as TUCS Research Report No 208 (November 1998), `http://www.tucs.fi`

26. Păun, G.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics 6, 75–90 (2001)
27. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Natural Computing 2(3), 265–285 (2003)
28. Pescini, D., Besozzi, D., Mauri, G., Zandron, C.: Dynamical probabilistic P systems. Intern. J. Found. Computer Sci. 17, 183–204 (2006)
29. Porreca, A.E., Mauri, G., Zandron, C.: Complexity classes for membrane systems. Informatique Theorique et Applications 40, 141–162 (2006)
30. Romero-Campero, F.J., Pérez-Jiménez, M.J.: Modelling gene expression control using P systems: the Lac Operon, a case study. BioSystems 91, 438–457 (2008)
31. Romero-Campero, F.J., Pérez-Jiménez, M.J.: A model of the quorum-sensing system in Vibrio fischeri using P systems. Artificial Life 14, 1–15 (2008)
32. Sosik, P.: The computational power of cell division. Natural Computing 2, 287–298 (2003)
33. Ciobanu, G., Păun, G., Pérez-Jiménez, M.J. (eds.): Applications of Membrane Computing. Springer, Heidelberg (2005)
34. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford (2010)
35. `http://www.p-lingua.org/`

Protecting Critical Infrastructures While Preserving Each Organization's Autonomy

Yves Deswarte[1,2]

[1] CNRS, LAAS, 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse, UPS, INSA, INP, ISAE, LAAS, F-31077 Toulouse, France
Yves.Deswarte@laas.fr

Abstract. In critical infrastructures (CIs), different organizations must cooperate, while being mutually suspicious since they have different interests and can be in competition on some markets. Moreover, in most cases, there is no recognized authority that can impose global security rules to all participating organizations. In such a context, it is difficult to apply good security practices to the interconnected information systems that control the critical infrastructure. In this paper, we present the PolyOrBAC security framework, aimed at securing global infrastructures while preserving each participating organization's autonomy. In this framework, each organization is able to protect its assets by defining its own security policy and enforcing it by its own security mechanisms, and the global infrastructure is protected by controlling and auditing all interactions between participating organizations. PolyOrBAC helps to satisfy the CII security requirements related to secure cooperation, autonomy and confidentiality, monitoring and audit, and scalability.

Keywords: Critical Infrastructure Protection, Security, Access Control Policies and Models, Collaboration, Interoperability.

1 Introduction

Our way of life relies on utilities provided by many Critical Infrastructures (CIs), such as those dedicated to electricity generation, transport and distribution (i.e., the electric power grid), telecommunications, supply services (energy, food, fuel, water, gas), transportation systems (whether by road, rail, air or sea), financial services (banks, stock exchange, insurances), etc. These infrastructures are critical, since their failure or disruption could potentially have a dramatic impact on economic and social welfare or health of a large population. Each of these infrastructures is controlled by an underlying information infrastructure, made of interconnected information and communication systems, including SCADA systems and other management information systems for operators, brokers, customers, etc. This information infrastructure is as critical as the infrastructure it controls, thus being a Critical Information Infrastructure (CII).

Public attention has been recently drawn on the related risks by the discovery of the Stuxnet worm targeting some SCADA systems[1].

Due to interdependencies between various infrastructures, cascading failures[2] and escalating failures[3] are not unlikely [33], [23], and a simple failure can propagate at a large scale, as in the case of the North America blackout that occurred on 14 august 2003 [8]. One of the immediate events that provoked this large blackout, with a cost estimated at between 7 and 14 billion dollars [21], was the failure of a monitoring software, which prevented confining an electrical line incident before it propagated across the electrical power grid. Such failure scenarios might occur as a result of accidental events as well as of malicious acts at the level of the infrastructure itself or of the information infrastructure (intrusions, worms and viruses, denials of service, etc.). Ensuring the security of CIIs is thus of tremendous importance, since a security breach in a CII can have catastrophic consequences.

As any other information system, a CII can be secured only by the combination of security policies, enforcement mechanisms, and monitoring and audit means. Security policies are specified as (1) security properties that the information systems must respect, and (2) rules that must be enforced to achieve and maintain these properties. These rules are to be enforced by authentication, access control and other protection mechanisms, including encryption for confidentiality, signatures for integrity, redundancy for availability, etc. And since policies can be imperfect or imperfectly enforced, monitoring and audit are needed to detect security violations, evaluate the damages and recover nominal operations, while collecting evidence to eliminate the exploited vulnerabilities and punish the culprits.

CIIs are more complex than most other information infrastructures. They are generally huge, extending over large geographic areas, and they interconnect very heterogeneous organizations, from individual users to multinational corporations. They must be flexible and extensible to incorporate new organizations, possibly over new geographical areas. Consequently, a CII must be open and distributed to allow the organizations participating in it to collaborate and provide globally the needed resources and services to their users. With the opening and the deregulation of markets, some of these organizations can be in competition, while they need to cooperate. The European electric grid is a typical instance of such a situation, where local, national and multinational companies are in competition but must cooperate to produce, transport and distribute electric power. Securing such an infrastructure is thus an extremely challenging task. To help in this task, this paper proposes a general framework to secure CIIs, taking into account their peculiarities, including scalability, flexibility and autonomy requirements.

---

[1] See for instance
`http://www.enisa.europa.eu/media/press-releases/stuxnet-analysis`

[2] Cascading failures occur when a failure in one infrastructure causes the failure of one or more components in a second infrastructure [23].

[3] Escalating failures occur when an existing failure in one infrastructure exacerbates an independent failure in another infrastructure, increasing its severity or the time for recovery and restoration from this failure [23].

The remainder of this paper is organized as follows: Section 2 identifies the generic security requirements of CIIs and confront these requirements to traditional access control models. Then a sketch of our proposal, the PolyOrBAC security framework, is presented in Section 3 and we show how this framework satisfies the CII security requirements. Section 4 presents briefly an experiment of PolyOrBAC application and gives some lessons learnt form it. Finally, Section 5 draws up the conclusions and proposes possible extensions of this work.

2 Problem Statement and Related Work

2.1 CII Security Requirements

Globally, a CI can be seen as a set of interacting organizations involving different actors and stakeholders (e.g., power generation companies, electric power transmission and distribution operators, energy brokers, national and transnational authorities, maintenance service providers, etc.). Such a CI is controlled through heterogeneous logical and physical information and communication systems and networks, exhibiting different levels of security threats and protection mechanisms.

The corresponding Critical Information Infrastructure (CII) can thus be viewed as a set of Local Area Networks (LANs) interconnected through a Wide Area Network (WAN) by dedicated switches and firewalls, noted CIS in Fig. 1 [38]. Each logical LAN is considered as a separate organization, composed of logical and physical information and communication systems, with its own applications and access control policy, which proposes its services to other systems. Each logical LAN belongs to a facility (e.g., power plant, substation, control center, etc.), and the WAN interconnects all the facilities belonging to the Critical Infrastructure. The CII is managed and accessed by different actors and stakeholders (e.g., power generation, transmission and distribution companies, regulation authorities, communication and computing system providers, brokers, subcontractors, etc.). Each logical LAN is dedicated to a CI component (e.g., a plant), in order to manage a different access control policy for each component. More than one LAN segment can be connected by the same CIS if they are part of the same organization and located in the same area.

In most cases, there is no global, recognized authority that can impose the same rules to all organizations participating in the infrastructure. Most often these organizations must reach a common agreement on how to operate the global infrastructure and on the interfaces and protocols through which their information systems should interact. And reaching such an agreement is not an easy task since these organizations are competing on the same markets, and thus have different interests, quite often conflicting, raising mutual suspicion. Moreover, they often operate legacy systems, not designed for cooperation, and they do not want to loose the control on their information systems, in particular for private tasks, independent of the infrastructure. Concerning the security policy, even if the participating organizations can agree on common security properties for the infrastructure, they would not easily accept to obey global rules, imposed

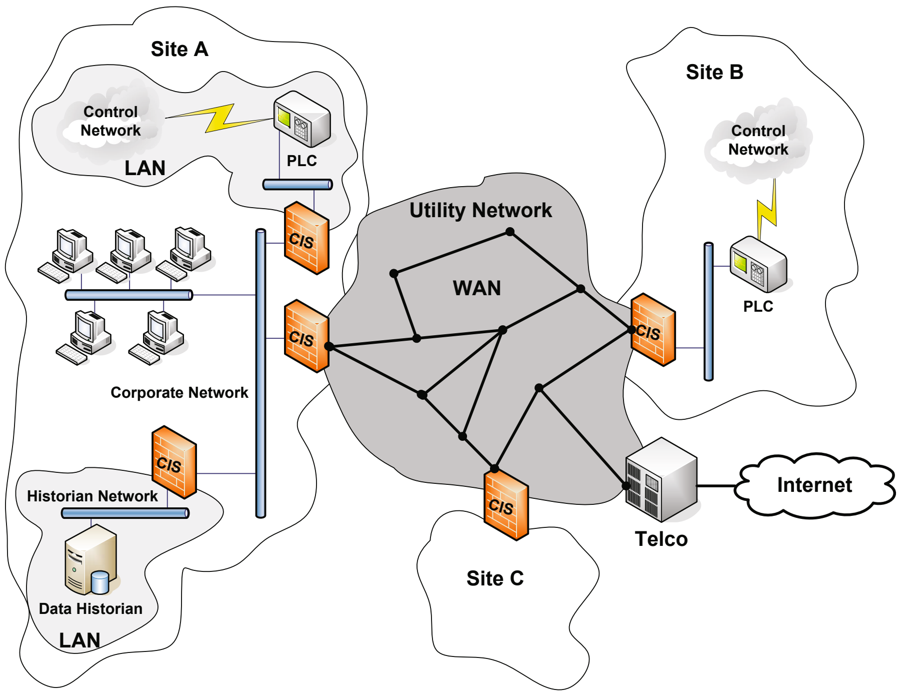


Fig. 1. General Architecture of a CII

by other partners, or on the mechanisms to enforce them. And finally, each organization wishes to protect its own business secrets as well as the privacy and autonomy of its personnel and of its assets, and would not wish to disclose more of its internal structure than needed for the infrastructure to operate safely. Concerning security, each organization wishes to be autonomous and protect its own assets by applying its own rules, enforced by its own mechanisms and procedures. The consequence of this autonomy is that each organization accepts, at least implicitly, its liability for the damages that any of its personnel may cause to other participating organizations.

These general constraints must be translated in the following requirements for the underlying Critical Information Infrastructure:

1. *Secure cooperation* between the information systems of the participating organizations, with different features, procedures and policies. In this context, it is necessary to control not only intra-organizational accesses (i.e., by a user of an organization to an object belonging to the same organization) but also interactions between different independent organizations.
2. *Autonomy, confidentiality and responsibility*: each organization controls its own security policy, users, resources, applications, etc., while respecting the global operation and security of the whole infrastructure, and maintains the confidentiality of its users and assets with respect to the other organizations,

as much as possible. On the other hand, an organization is responsible and liable for all actions that it authorizes its users to perform to the other organizations.

3. *Monitoring and audit*: especially for inter-organizational workflows, monitoring should determine if the rules applying to interactions between organizations are correctly enforced in practice and audit should keep logs on interactions between partners, to provide evidence in case of dispute or abuse.
4. *Scalability*: the security framework should be easily adaptable to large, evolving infrastructures, where organizations can dynamically be integrated or leave, without requiring too much change in the security policies and their enforcement.

2.2 Related Work

To satisfy the CII security requirements cited above, two global approaches can be contemplated: centralized or peer-to-peer. In the following, we discuss some examples of significant contributions that range between these two extreme approaches.

Centralized approaches. Several works on Workflow Management Systems follow the centralized approach. For example, Bertino *et al.* describe different configurations and constraints associated to workflow execution [12]. Adam *et al.* use colored and timed petri nets to define a conceptual and a logical workflow authorization model based on the inter-dependencies between activities [6]. However, these works do not show how to enforce inter-organization workflows while respecting the global and local constraints, especially when organizations collaborate to achieve a common objective. Moreover, these works suppose the existence of a central entity responsible for specifying and managing the workflow security policy without describing how to enforce such a policy. Hence, applied to our context, these solutions would require the definition of a global security policy for the CII, according to which all participating organizations must adapt their own security policies. Finally, these solutions do not provide a framework for the security policy to be dynamically monitored during the workflow execution, while this requirement is important in our context. Indeed, such a centralized approach is incompatible with the autonomy requirements of most CIIs.

Lin *et al.* [25] proposed a novel access control model for collaborative organizations, based on policy decomposition. This architecture is based on the XACML framework [30], which allows the proposed solution to be easily integrated into existing systems. It also presents algorithms for decomposing a global policy that is enforced by a set of collaborating parties without compromising the autonomy or confidentiality requirements of the collaborating parties and to efficiently evaluate requests from different parties. While very interesting, this work cannot be directly applied in our context as it imposes a global access control policy over the whole collaborative environment, which is not easily compatible with our autonomy, confidentiality and scalability requirements.

Bertino, Jajodia and Samarati [13] presented a unified framework that can enforce multiple access control policies within a single system. The framework is based on a language through which users can specify security policies to be enforced on specific accesses. The language allows the specification of both positive and negative authorizations and incorporates notions of authorization derivation, conflict resolution, and decision strategies. Different strategies may be applied to different users, groups, objects, or roles, according to the needs of the security policy. The major advantage of this approach is that it can be used to specify different access control policies that can be enforced by the same security server. However it does not take into consideration collaboration and autonomy issues, which are important in the context of CIIs.

In term of languages, other significant contributions are related to Ponder [17] and XACML. For example, Lorch *et al.* propose to use XACML as one component of a distributed and inter-operable authorization framework [26]. This work illustrates how authorization can be deployed in distributed, decentralized systems, and helps connecting the general components of an authorization system. XACML is useful for specifying complex policies in a wide variety of distributed applications, environments and systems. However, the language flexibility and expressiveness comes at the cost of complexity and verbosity. In practice, using this language is painful as it leads to complex policy description files. Tools are underway, but as long as they are not widely available, it will be hard for average users to work with any XACML-based system. And even with good tools in place, there is an inherent semantic complexity that accumulates over the syntactic complications.

Peer-to-Peer approaches. Peer-to-peer solutions have been proposed in some previous works. These approaches do not assume the existence of a global CII organization. In 2008, Sturm *et al.* [36] presented a fine grained access control mechanism for peer-to-peer collaborations. This mechanism is based on the local access control policies of the participants, expressed in XACML and exported to the other participating organizations. Two methods are proposed to combine these policies. The first one consists in establishing mappings between exported policies. The second method consists in installing a distributed access control directory. While mappings are created between two peers (at the price of disclosing significant private information to the other peer), a directory contains all rights of all users of all peers of all the participating organizations. Both methods are thus unsatisfactory with respect to the confidentiality requirement.

In a similar way, Shehab *et al.* [35] presented a distributed secure interoperability framework for mediator-free collaboration environments in which domains collaborate in making localized access control decisions. This work introduced the idea of secure access paths which enables domains to take local access control decisions without having a global view of the collaboration. It also presents a path authentication technique for proving path authenticity. Basically, this uses proactive and on-demand path discovery algorithms that enable domains to securely discover paths in the collaboration environment. This technique can establish secure channels for interactions between organizations, but does not help in controlling whether these interactions are compatible with global security rules.

Pearlman *et al.* [32] presented the Community Authorization Service (CAS) intended to solve three critical authorization problems that arise in distributed virtual organizations: scalability, flexibility and expressibility, and the need for policy hierarchies. Their work addressed these problems by introducing a trusted third party administrated by the virtual organization that performs fine-grain control of community policy while leaving ultimate control of resource access to the responsibility of resource owners. This work is interesting in the fact that it presents an authorization framework for distributed and collaborative environments. However, it requires a third party with a global knowledge of policies of other organizations. This is a limitation that our work tries to overcome.

In a similar way, MultiOrBAC [2] and O2O [16] stipulate that the various organizations accept to cooperate so that roles in one organization are given privileges in another organization. For that, each participating organization must trust the others, at least for the definition of some of their roles and for the assignment of the corresponding roles to trustworthy users. This approach is also intrusive with respect to the confidentiality of each organization's internal structure, user identity, and security policy. This is equally unacceptable in our context, where each organization wants to keep its autonomy on the choice of its internal security policy, and would not accept to open its information and communication system to unknown external users working for its competitors. Ideally, an organization should know nothing about the other organizations' users or assets, but only the information needed to cooperate fairly. Enabling a secure collaboration between organizations while preserving each organization's autonomy and self-determination is the challenge addressed by our approach, the PolyOrBAC framework, presented in the next section.

3 The PolyOrBAC Security Framework

With PolyOrBAC [5], we intend to provide a global security framework, where each organization participating in a Critical Infrastructure is autonomous for protecting its assets by defining its own security policy and enforcing it by its own security mechanisms. To interact securely with its partners, the organization integrates in its security policy the access points to the other organizations it interacts with, and control local accesses to these access points in the same way as for other local objects. On the other hand, the organization provides also access points to other organizations it interacts with, and controls the actions preformed by the other organizations through these access points in the same way as it controls the actions made by its own users. In this framework, interactions between organizations are limited to Web Services [3], and for each Web Service, the service provider and the service client must have previously signed a contract that defines the conditions of service provision and the rules governing the interactions. All interactions are logged by both the service provider and the service client, which can present them to a judge in case of dispute or abuse, with reference to the signed contract. This framework is intended to satisfy all the above constraints of (1) secure cooperation, (2) autonomy, confidentiality and responsibility, (3) monitoring and audit, and (4) scalability.

PolyOrBAC defines, deploys and audits a security framework for both intra- and inter-organizational workflows. It mainly gives answers to questions such as: how to define intra-organizational security policies? how to specify inter-organizational access policies? how to specify and deploy e-contracts that can be agreed between organizations collaborating within a CII? how to enforce access control, and how to detect and audit possible violations and abuses at runtime? The following subsections describe PolyOrBAC proposal: first, we show how to specify local security policies within each organization, based on the OrBAC model [1]; then, we introduce new notions (virtual users and WSs images) to manage inter-organizational accesses; and finally, we define e-contracts to express and check WSs interactions at runtime.

3.1 Specifying Local Security Policies with OrBAC

In PolyOrBAC, each organization specifies its own security policy, which defines which user has access to what, when, and in which conditions. In this subsection, we show that OrBAC is a suitable access control model for achieving this task. First, let us recall the main notions of OrBAC and discuss them with respect to some other access control models.

The OrBAC (*Organization-based Access Control*) model is an extension of the traditional RBAC (*Role-Based Access Control*) model [34], [19]. In RBAC, roles are assigned to users, permissions are assigned to roles and users acquire permissions by playing roles. By abstracting users into roles, RBAC facilitates security management: if users are added to or are withdrawn from the system, only instances of the relationship between users and roles need to be updated. OrBAC goes further by abstracting objects into views and actions into activities. In this way, security rules are specified by abstract entities only, and the representation of the security policy is completely separated from the implementation.

More precisely, in OrBAC, an activity is a group of one or more actions; a view is a group of one or more objects; and each rule expresses if an authorization, prohibition or obligation applies for a role to perform an activity on a view in a certain context. Actually, two levels can be distinguished in OrBAC:

– *Abstract level*: the organization's security administrator defines rules by using abstract entities (roles, activities, views) without worrying about how the organization implements these entities.
– *Concrete level*: when a user requests to perform an action on an object, permissions are granted to him according to the concerned rules, the role currently played by the user, the requested action (that instantiates an activity defined in the rule) on the object (that instantiates a view defined in the rule), and the current context.

The derivation of permissions (i.e., runtime evaluation of security rules) can be formally expressed as follows:

$\forall$ org $\in$ Organizations, $\forall$s $\in$ Subjects, $\forall$ α $\in$ Actions, $\forall$ o $\in$ Objects, $\forall$ r $\in$ Roles, $\forall$a $\in$ Activities, $\forall$ v $\in$ Views, $\forall$ c $\in$ Contexts
***Permission* (org, r, v, a, c)** $\wedge$
Empower (org, s, r) $\wedge$
Consider (org, α, a) $\wedge$
Use (org, o, v) $\wedge$
Hold (org, s, a, o, c)
$\rightarrow$ ***Is permitted*(s, α, o)**

This rule means: if in a certain organization *org*, a security rule specifies that role *r* can carry out the activity *a* on the view *v* when the context *c* is *true*, and if *r* is assigned to subject *s*, if action α is a part of *a*, and if object *o* is part of *v*, and if *c* is *true*, then *s* is allowed to perform α (e.g., WRITE) on *o* (e.g., F1.TXT). Prohibitions and obligations can be defined in the same way.

As rules are expressed only through abstract entities, OrBAC is able to specify the security policies of several collaborating and heterogeneous sub-organizations (e.g., departments) of a "global organization". In fact, the same role (e.g., OPERATOR) can be played by several users belonging to different sub-organizations; the same view (e.g., "TECHNICALFILE"), can designate a table TF-TABLE in one sub-organization or a XML object TF1.XML in another one; and the same activity READ can correspond in a particular sub-organization to a SELECT action while in another sub-organization it may specify an OPENXMLFILE() action.

In our context, OrBAC presents several benefits and satisfies several security requirements of organizations participating in a CII: rule expressiveness, abstraction of the security policy, scalability, heterogeneity and evolvability. OrBAC is thus more suitable than RBAC (and other variants), in particular for specifying local security policies of the CII's organizations. These security policies can subsequently be locally enforced by the security mechanisms implemented by the local organization, e.g., Access Control Lists (ACL), firewall rules, security credentials (e.g., XML capabilities), OASIS WS security mechanisms, etc.

3.2 Managing Interactions between Organizations

While OrBAC is suitable for specifying local security policies, it suffers a limitation that is important in our context: OrBAC is limited to the specification of a single security policy and does not handle collaborations between autonomous organizations having independent policies. In fact, an OrBAC policy belonging to a given organization cannot specify rules that associate permissions to users belonging to other organizations or to control access to resources belonging to other organizations. As a result, OrBAC is unfortunately only adapted to infrastructures with a global security policy and thus does not cover the distribution and collaboration needs of current CIIs presented in Section 2.1.

As an attempt to fulfill these needs, we first proposed the MultiOrBAC model in [2]. Basically, MultiOrBAC abstract rules specify that roles in a certain organization are permitted (or prohibited or obliged) to carry out activities on views belonging to other organizations. Therefore, contrarily to OrBAC, a MultiOrBAC rule may involve two different organizations that do not belong to the same hierarchy: the organization where the role is played, and the organization which the view and the activity belong to. However, in the context of CIIs, MultiOrBAC presents several weaknesses. In fact, MultiOrBAC offers the possibility to define local rules that control accesses to local objects from external roles (i.e., belonging to another organization), without having any information on who plays these roles and how the (*user*, *role*) association is managed by the remote organization. This causes a serious problem of responsibility and liability: who is responsible in case of remote abuse of privileges? how can the organization to which belongs the object trust the organization to which belongs the user? MultiOrBAC logic is thus not adapted to CIIs, where in-competition organizations are mutually suspicious. Moreover, in MultiOrBAC access control decision and enforcement are done independently by each organization, which means that a global security policy is in fact defined by the set of the organizations' security policies. In that case, it is difficult to enforce and maintain the consistency of the global security policy, in particular if each organization's security policy evolves independently.

In the PolyOrBAC framework, collaboration and interactions between organizations are made through the use of the Web Service technology (WS), which provides platform-independent protocols and standards for exchanging heterogeneous interoperable services. Software applications written in various programming languages and running on various platforms can use WS to exchange data over computer networks in a manner similar to inter-process communication on a single computer. WS also provide a common infrastructure and services for data access, integration, provisioning, cataloging and security [29]. These functionalities are made possible through the use of open standards, such as: XML for exchanging heterogeneous data in a common information format [31]; SOAP, a protocol to exchange data between different applications running on one or several operating systems [40]; WSDL, used to describe the services that a business offers and to provide a way for individuals and other businesses to access those services [41] and UDDI, an XML-based registry which enables businesses to list themselves and publish their services (in WSDL) on the Internet and discover each other [28].

Note that some recent works already tried to combine web services mechanisms and security policies based on RBAC. Beznosov and Deng presented a framework for implementing Role-Based Access Control using CORBA security service [15]. Vuong, Smith and Deng proposed an XML-Based approach to specify enterprise RBAC policies [39]. In 2004, Feng, Guoyuan and Xuzhou suggested SRBAC, a Service-oriented Role-Based Access Control model and security architecture model for Web Services [18]; Leune and van den Heuvel presented RBAC4WS, a methodology for designing and developing a Role-Based

Access Control model for Web Services [24]. Focusing on service invocation, this methodology adopts a symmetric perspective considering both the supplier and the customer. Besides, some other works tried to couple XACML with RBAC. For example, in 2004, OASIS adopted an XACML profile for Role Based Access Control, while in 2005, Crampton proposed an RBAC policy using an XACML formulation [30].

In the proposed PolyOrBAC framework, we integrate WS and OrBAC. To achieve this task, we introduce two new notions, *virtual users* and *WS images*:

– for the organization offering a WS (i.e., that allows external accesses to its local resources through a WS interface), the client organization is seen as a *virtual user* which plays a role authorized to use the WS;
– for the organization requesting the WS, the WS is seen as an external object, locally represented by its *WS image*.

To illustrate these notions, let us describe the two main phases of PolyOrBAC: (1) publication and negotiation of collaboration rules, including the corresponding access control rules, and (2) runtime access to remote services.

In the *first phase*, each organization determines which resources it will offer to other partners. Web services are then developed on application servers, and published in a UDDI registry to be accessible to external users.

When an organization has published its WS in the UDDI registry, the other organizations can contact it to express their wish to use the WS. Let us take a simple example where organization B offers WS1, and organization A is interested in using WS1. A and B should negotiate and come to an agreement concerning the use of WS1. Then, A and B establish a contract[4] and jointly define security rules concerning the access to WS1. These rules are registered (according to an OrBAC format) in databases located at both A and B (typically in their CIS, see Section 3.1). For instance, if the agreement between A and B is "*users from A have the permission to consult B's measurements in the emergency context*", B should, in its OrBAC security policy:

– have (or create) a rule that grants the permission to a certain (local) role (e.g., OPERATOR) to consult its measurements: *Permission*(*B*, *Operator*, *Measurements*, *Consult*, *Emergency*);
– create a VIRTUAL USER noted *PartnerA* that represents A for its use of WS1;
– add the *Empower*(*B*, *PartnerA*, *Operator*) association to its rule base. By this rule, organization *B* grants *PartnerA* the right to play the OPERATOR role.

In parallel, A creates locally a *WS1_image* which (locally in A) represents WS1 (i.e., the WS offered by B), and adds a rule in its OrBAC base to define which of A's roles can run the action *invoke* on object *WS1_image* to use WS1.

Considering the *second phase* of PolyOrBAC dedicated to the control of runtime access to remote services, we use an AAA (*Authentication*, *Authorization*

[4] The contract aspects will be discussed in the next subsection.

and *Accounting*) architecture, which separates authentication from authorization; we distinguish access control decision from access control enforcement; and we keep access logs in each organization. Basically, if a user from A (let us note it Alice) wants to carry out an activity, she is first authenticated by A. Then, protection mechanisms of A check if the OrBAC security policy (of A) allows this activity. We suppose that this activity contains local as well as external accesses (e.g., invocation of B's WS1). Local accesses should be controlled according to A's policy, while the WS1 invocation is both controlled by A's policy (Alice must play a role that is permitted to run the action *invoke* on object *WS1_image*), and by B's policy (the invocation is transmitted to virtual user *PartnerA*, which must play a role authorized to execute the web service), according to the contract established between A and B. If both policies grant the invocation, WS1 is executed (under the access control enforcement mechanisms implemented by A and by B).

3.3 Expressing and Checking WS Interactions with e-Contracts

In the previous subsection, we have shown that PolyOrBAC offers several useful concepts and mechanisms for access control in CIIs: it permits a better specification and control of local security policies through OrBAC; each organization authenticates its users and manages its resources autonomously; and interactions are handled by WS. Consequently, the service-requesting organization is liable for its users, and thus is responsible for the actions carried out by their users. In the same way, the service-providing organization is liable for the services it offers. However, other aspects need to be addressed:

– Enforcement and real time checking of contracts established between different organizations; in fact, the system must be able to check the satisfaction as well as the correct enforcement of the signed contracts.
– Audit logging and assessment of the different actions: in fact, in large scale systems, experience has shown that even if the security policy is consistent (thanks to an off-line verification), violations and abuses (especially, remote abuse of privileges) can occur dynamically at runtime. Hence, we need a mechanism that can detect this kind of dysfunctional behavior and to notify the concerned parties.
– Handling of mutual suspicion between organizations: no information is disclosed about local security policies and the organization providing the web service does not know which user of the other organization requests the web service, and the organization requesting the web service does not know which role performs which activity on which views in the service providing organization. It is also necessary to detect any abuse of the contract by a malicious organization.

To deal with these issues, we state that for each WS use, an e-contract should be negotiated between the two partner organizations (the WS provider and the WS client). This contract must specify precisely the web service functions and parameters (including the expected quality of service, the liability of each party, payment for service use, penalties in case of abuse, etc.), and also the security rules related to the invocation and the result provision of the web service. These security rules must be checked and enforced at runtime to prevent, or at least detect, any abuse. The security rules can be expressed with a syntax close to OrBAC (see Section 3.1).

The question that arises now is how to specify e-contracts. Actually, the most relevant security notions for such e-contracts are workflows, actions, permissions, prohibitions, obligations, time constraints, disputes and sanctions.

To express these requirements, we propose using *timed automata* [7]. First, permissions (actions that are authorized by the contract clauses) are simply specified through transitions in the timed automata. For instance, in Fig. 2, the system can (i.e., has the permission to) execute the action a at any time and then, behaves like the automaton A.

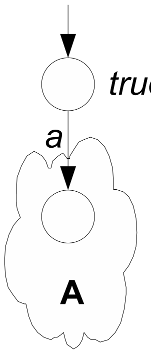


Fig. 2. Modeling Permissions

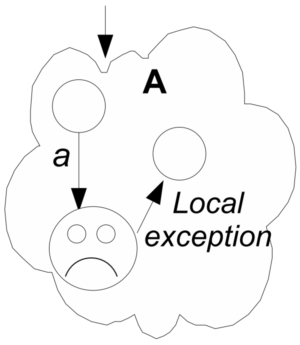


Fig. 3. Modeling prohibitions

Second, we distinguish two kinds of prohibitions in e-contracts:

- *Implicit prohibitions*: the idea is that permissions being transitions in the automata, the states, actions and transitions not represented in the automata are by essence *prohibited*, so the runtime model checker will not recognize them, and thus will halt the execution.
- *Explicit prohibitions*: explicit prohibitions can be particularly useful in the management of decentralized policies / contracts where an organization's security administrator does not have details about the other organizations participating in the CII, and thus has to trigger exception procedures in case of unauthorized interactions. Moreover, explicit prohibitions can also limit the propagation of permissions in case of hierarchies. In our model, we specify explicit prohibitions by adding a "failure state" where the system will be automatically led if a malicious action is detected. In Fig. 3, as the a action is forbidden, its execution automatically leads to the failure state described by an "unhappy face", which automatically triggers an exception carried out locally.

Let us now deal with obligations. Recently, several works have focussed on the modeling of this access modality [14] [17] [27] [22]. In XACML [30], obligations are a set of operations that must be fulfilled in conjunction with an authorization decision (permit or deny). Bettini *et al.* distinguish between provisions and obligations [14]. Provisions are conditions that need to be satisfied or actions that must be performed before a decision is rendered, while obligations are actions that must be fulfilled by either the users or the system after the decision. Hilty *et al.* define an Obligation Specification Language (OSL), that allows formulating a wide range of usage control requirements [22]. They differentiate between usage and obligational formulae. Usage is concerned with operations (e.g., processing, rendering, execution, management, or distribution) on data that must be protected; while obligational formulae are conditions on the usage of data, e.g., "delete document D within 30 days". An obligational formula becomes an obligation once a data consumer is obliged to satisfy it, i.e., once the data consumer has received the data and committed to the condition.

In our vision, obligations are actions that must have been carried out before a specified event occurs (e.g., before a given delay); otherwise the responsible entity will be subject to sanctions. Besides that, as every obligation is also a permission[5], obligations will be specified by particular transitions (in the same way as permissions). However, as obligations are stronger than permissions, we should add another symbols to capture this semantics and to distinguish between what is mandatory and what is permitted but not mandatory. Actually, to model obligations, we use transition time-outs and invariants.

In this respect, an obligation is considered as a simple transition, and if a maximum delay is assigned to the obligation, a *time-out* (noted by d in Fig. 4) is set for the delay. When the obligation is fulfilled, this event resets the time-out and the system behaves like A1. On the contrary, if the time-out expires, an exception is raised and the system behaves like A2 (which can be considered as an *exception*).

Basically, when an explicit prohibition occurs because an obligation is not fulfilled, a dispute situation (e.g., one of the parties does not comply with the contract clauses) arises, and the automaton automatically makes a transition to a failure state or triggers an exception processing (like A2 in Fig. 4). Actually, modeling disputes will allow to not only identify anomalies and violations, but go further by identifying activities (sequence of actions and interactions) that led to these situations, and finally can automatically lead to the cancelation of the contract. Moreover, as disputes have different severities and as they are not all subject to the same sanctions, we use variables (i.e., labels on the failure states) to distinguish the different kinds of disputes as well as the corresponding sanctions (Fig. 5).

Note that once the expected behaviors of the contracting parties are modeled by timed automata, it is possible to verify some security properties statically, and then enforce them at run-time by checking the execution of the system

[5] A mandatory action should be permitted; in other words, we cannot render mandatory something that is not permitted.

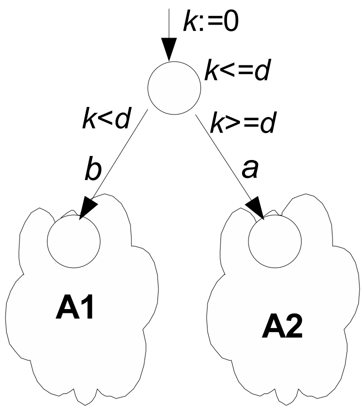


Fig. 4. Modeling obligations

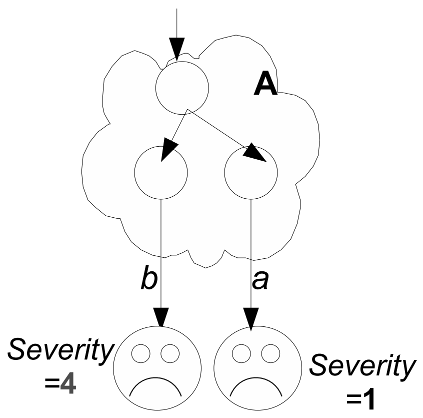


Fig. 5. Modeling dispute situations

dynamically against the behaviors specified by the model [4]: by static analysis, we can (1) check e.g. under which conditions the system can reach a dispute state; at runtime, we can (2) maintain an audit log and perform model-checking by monitoring the interactions and mapping them to the timed-automata; and (3) notify the concerned parties in case of contract violation (i.e., when a failure state is reached).

This is important in our context where the collaborating organizations are in mutual suspicion. In practice, the security policy of the service-providing organization discards any request that does not correspond to a signed contract. This is a first level of security. The second level is enforced by runtime model checking. Hence, even if a user succeeds in bypassing his own organization's security policy and interacts in a wrong way (by accident or by malice) with another organization (e.g., by using an authorized Web service in a way which is in contradiction with the signed contract rules), the forbidden interactions are detected and audited (i.e., logged) at runtime. In the same way, if the providing organization does not satisfy its obligations, our e-contract model checking will detect and audit the corresponding misbehavior.

4 Experiment and Lesson Learnt

In a demonstration experiment, we successfully applied the PolyOrBAC framework to an emergency scenario of a smart electric power grid [20], where the transport operator detects a risk of overload that could lead to a blackout, then alerts a distribution operator to prepare for shedding a given load if the critical situation occurs. The distribution operator then selects which distribution substations must be armed to shed a sufficient load while minimizing the risks for its customers. Then if the overload is detected by a transport substation, this substation will automatically send a load shedding command to all concerned distribution substations, and then the armed substations will open their circuits in a few milliseconds, thus avoiding the blackout.

This experiment has been implemented on a network of four computers, running up to twenty virtual machines to simulate the transport control center with its console, the distribution control center with its console, a transport substation and four distribution workstations. Each organization (transport control center, distribution control center, and each substation) was composed of two virtual machines, one simulating the corresponding information system, and another one corresponding to the CIS connecting this organization to the network, and responsible for model-checking the interactions with the other organizations. The behavior of the electric grid was simulated with realistic production and transport characteristics as well as variable power consumption at the distribution substation level, under the control of an experiment management system and its console. Various situations have been emulated, nominal as well as emergency situations, without and with realistic attacks, injected at various points of the simulated CII. More information on this experiment are given in [9], while [5] presents a summary of the scenario and some details on its implementation.

The first lesson learnt form this experiment is that it has been relatively easy to implement all interactions between the various organizations (transport control center, distribution control center, and each substation) by means of Web Services, while this is not the traditional way to implement such complex control systems. For each interaction, all the Web Service exchanges were first modeled by timed automata, and these automata were checked at runtime by a model checker implemented with UPPAAL [37] in each concerned CIS. The efficiency of these mechanisms has been confirmed by our attack experiments. Each organization had its own security policy, with its roles, activities and views, corresponding to "real" users (e.g., transport and distribution operators) with their privileges, as well as virtual users corresponding to the Web Services. Attacks simulating intrusions in the various organizations (including attacks by malicious operators and by outsiders attacking the network) have been simulated to observe their effect on local access control enforcement and on CIS Web Service interaction monitoring and exception handling. In no case, an intrusion into one organization could disturb another organization without being detected and recovered by at least one CIS. Even a fake initial action by a malicious transport operator (i.e., a critical situation alert sending to the distribution operator) would have no consequence on the other organizations. But of course, an alert omission by a malicious transport operator can result in a larger than necessary load shedding, if both a power grid overload occurs and no other safety countermeasure is in place. Denial of service attacks could have the same effect, but have been addressed by another study within the same European project [10].

The second lesson learnt from this experiment is that PolyOrBAC is indeed able to fulfill the CII security requirements:

1. *Secure cooperation* between the information systems of the participating organizations, has been experimented successfully.

2. *Autonomy, confidentiality and responsibility*: a specific security policy has been defined for each organization, and the only information known by an organization about other ones where limited to the Web Service characteristics. Each organization controls all actions performed by its users, and thus is responsible for their acts.
3. *Monitoring and audit*: the Web Service interactions were successfully monitored and audited, and when an abuse was detected, enough logs had been collected to identify the organization responsible for the abuse.
4. *Scalability*: since each organization knows only those organizations that either provides it a service or requests one of its services, the number of contracts and model-checking automata grows linearly (with a small coefficient) with the number of organizations instead of with the square of this number. Moreover, at least in our experiment, the complexity of the timed automata specifying the Web Service interactions has proven to be very low (see [5] for examples of such automata).

5 Conclusion

In this paper, the PolyOrBAC security framework has been presented, which meets access control and collaboration requirements of CIIs. Even if several works had previously investigated security in workflow and collaborative systems, none of these works have defined an homogeneous peer-to-peer approach going from the specification to the deployment and runtime model-checking. Moreover, none of them have been applied to secure CIIs. Dealing with these issues, the PolyOrBAC framework manages collaboration and resources sharing between all organizations of a CII thanks to the web services technology, while controlling that the interactions between these organizations are in conformity with their needs and their internal security policies specified with OrBAC. PolyOrBAC supports the enforcement, the real-time checking as well as the auditing of the exchanges that are established between the different organizations participating in a CII. Even if we experimented it only in a particular critical infrastructure simulation, we are confident that our work can be successfully applied to most other critical infrastructures, and can also benefit to non-critical collaborative systems, especially those with mutually suspicious organizations.

Acknowledgments

This work has been partially supported by the European FP6-IST research project CRUTIAL (CRitical UTility InfrastructurAL Resilience)[6], the European Networks of Excellence ReSIST[7] and the LAAS project PolSec[8]. The author is grateful to all colleagues who participated in these projects, and more particlarly to those who contributed directly to the design, the development and the experiment of PolyOrBAC: Mohamed Kaâniche, Anas Abou El Kalam and Amine Baïna.

[6] http://crutial.erse-web.it/
[7] http://www2.laas.fr/RESIST/
[8] http://www2.laas.fr/PolSec/

References

1. Abou El Kalam, A., El Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization Based Access Control. In: Proc. of IEEE 4th Intl Workshop on Policies for Distributed Systems and Networks (POLICY 2003), Lake Come, Italy, June 14-16, pp. 120–131 (2003)
2. Abou El Kalam, A., Deswarte, Y.: Multi-OrBAC: a New Access Control Model for Distributed, Heterogeneous and Collaborative Systems. In: IEEE Symp. on Systems and Information Security (SSI 2006), Sao Paulo, Brazil (2006)
3. Abou El Kalam, A., Deswarte, Y., Baïna, A., Kaâniche, M.: Access Control for Collaborative Systems: A Web Services Based Approach. In: IEEE Intl Conf. on Web Services (ICWS 2007), Salt Lake City, Utah, USA, July 9-13, pp. 1064–1071 (2007)
4. Abou El Kalam, A., Deswarte, Y.: Critical Infrastructures Security Modeling, Enforcement and Runtime Checking. In: Setola, R., Geretshuber, S. (eds.) CRITIS 2008. LNCS, vol. 5508, pp. 95–108. Springer, Heidelberg (2009)
5. Abou El Kalam, A., Deswarte, Y., Baïna, A., Kaâniche, M.: PolyOrBAC: A Security Framework for Critical Infrastructures. International Journal of Critical Infrastructure Protection (IJCIP) 2, 154–169 (2009)
6. Adam, N.R., Atluri, V., Huang, W.-K.: Modeling and Analysis of Workflows Using Petri Nets. Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management 2(2), 131–158 (1998)
7. Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126(2), 183–235 (1994)
8. Amin, M.: North America's Electricity Infrastructure: Are We Ready for More Perfect Storms? IEEE Security and Privacy 1(5), 19–25 (2003)
9. Baïna, A.: Modèles et politiques de sécurité pour la protection des infrastructures critiques, Doctorate Thesis, Université de Toulouse, LAAS-CNRS (September 29, 2009) (in French)
10. Beitollahi, H., Deconinck, G.: An Overlay Protection Layer Against Denial-of-Service Attacks. In: 22nd IEEE Intl Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, Florida, May 14-18, pp. 1–8 (2008)
11. Berard, B., Bidiot, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P., McKenzie, P.: Systems and Software Verification, Model Checking Techniques and Tools. Springer, Heidelberg (2001) ISBN 3-540-41523-7
12. Bertino, E., Ferrari, E., Alturi, V.: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Transactions on Information and System Security (TISSEC) 2(1), 65–104 (1999)
13. Bertino, E., Jajodia, S., Samarati, P.: Flexible Support for Multiple Access Control Policies. ACM Transaction on Database Systems (TODS) 26(2), 214–260 (2001)
14. Bettini, C., Jajodia, S., Wang, X.S., Wijesekera, D.: Obligation Monitoring in Policy Management. In: Proc. of IEEE 3rd Intl Workshop on Policies for Distributed Systems and Networks (POLICY 2002), Monterey, CA, June 5-7, pp. 2–12 (2002)
15. Beznosov, K., Deng, Y.: A Framework for Implementing Role-Based Access Control Using CORBA Security Service. In: 4th ACM Workshop on Role-Based Access Control, Fairfax, VA, USA, October 28-29, pp. 19–30 (1999)
16. Cuppens, F., Cuppens-Boulahia, N., Coma, C.: O2O: Virtual Private Organizations to Manage Security Policy Interoperability. In: Bagchi, A., Atluri, V. (eds.) ICISS 2006. LNCS, vol. 4332, pp. 101–115. Springer, Heidelberg (2006)

17. Damianou, N., Dulay, N., Lupu, E.: The Ponder Policy Specification Language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001)
18. Feng, X., Guoyuan, L., Xuzhou, X.: Role-based Access Control System for Web Services. In: 4th International Conference on Computer and Information Technology (CIT 2004), Wuhan, China, September 14-16, pp. 357–362 (2004)
19. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security (TISSEC) 4(3), 224–274 (2001)
20. Garrone, F., Brasca, C., Cerotti, D., Codetta Raiteri, D., Daidone, A., Deconinck, G., Donatelli, S., Dondossola, G., Grandoni, F., Kaaniche, M., Rigole, T.: Analysis of new control applications. CRUTIAL project, Deliverable D2 (January 2007)
21. Hilt, D.W.: August 14, 2003, Northeast Blackout Impacts and Actions and the Energy Policy Act of 2005. In: North American Electric Reliability Council (NERC), Presentation at ISPE Annual Conference (August 2, 2006), `http://www.nerc.com/filez/blackout.html`
22. Hilty, M., Pretschner, A., Basin, D., Schaefer, C., Walter, T.: A Policy Language for Distributed Usage Control. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 531–546. Springer, Heidelberg (2007)
23. Laprie, J.C., Kanoun, K., Kaâniche, M.: Modelling Interdependencies Between the Electricity and Information Infrastructures. In: Saglietti, F., Oster, N. (eds.) SAFECOMP 2007. LNCS, vol. 4680, pp. 57–67. Springer, Heidelberg (2007)
24. Leune, K., van den Heuvel, W.-J.: A Methodology for Developing Role-Based Access/Control to Web-Services. Tilburg University, Infolab Technical Report Series, no. 11 (December 2002)
25. Lin, D., Rao, P., Bertino, E., Li, N., Lobo, J.: Policy Decomposition for Collaborative Access Control. In: 13th ACM Symposium on Access Control Models and Technologies (SACMAT 2008), Estes Park, CO, USA, pp. 103–112 (2008)
26. Lorch, M., Proctor, S., Lepro, R., Kafura, D., Shah, S.: First Experiences Using XACML for Access Control in Distributed Systems. In: 2003 ACM Workshop on XML Security, Fairfax, VA, pp. 25–37 (2003)
27. Ni, Q., Bertino, E., Lobo, J.: An Obligation model bridging access control policies and privacy policies. In: 13th ACM SACMAT, Estes Park, CO, USA, June 11-13 (2008)
28. OASIS, Universal Description, Discovery and Integration v3.0.2 (UDDI), UDDI Specification TC, OASIS Standard (February 2005)
29. OASIS, Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), OASIS Standard Specification (February 1, 2006)
30. OASIS, eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard (February 1, 2005)
31. OASIS, XML Catalogs, OASIS Standard V1.1 (October 7, 2005)
32. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A Community Authorization Service for Group Collaboration. In: Proc. of IEEE 3rd Intl Workshop on Policies for Distributed Systems and Networks (POLICY 2002), Monterey, CA, June 5-7, pp. 50–59 (2002)
33. Rinaldi, S.M., Peerenboom, J.P., Kelly, T.K.: Identifying, understanding, and analyzing critical infrastructure interdependencies. IEEE Control Systems Magazine 21(6), 11–25 (2001)
34. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-Based Access Control Models. IEEE Computer 29(2), 38–47 (1996)

35. Shehab, M., Bertino, E., Ghafoor, A.: Secure Collaboration in Mediator-Free Environments. In: 12th ACM Conference on Computer and Communications Security (CCS 2005), Alexandria, VA, pp. 58–67 (2005)
36. Sturm, C., Dittrich, K.R., Ziegler, P.: An access control mechanism for P2P collaborations. In: Proceedings of the 2008 International Workshop on Data Management in Peer-to-peer Systems (DaMaP 2008), Nantes, France, March 25, pp. 51–58 (2008)
37. UPPAAL tool available at, http://www.uppaal.com
38. Verissimo, P., Neves, N.F., Correia, M., Deswarte, Y., Abou El Kalam, A., Bondavalli, A., Daidone, A.: The CRUTIAL Architecture for Critical Information Infrastructures. In: de Lemos, R., Di Giandomenico, F., Gacek, C., Muccini, H., Vieira, M. (eds.) Architecting Dependable Systems V. LNCS, vol. 5135, pp. 1–27. Springer, Heidelberg (2008)
39. Vuong, N., Smith, G.S., Deng, Y.: Managing Security Policies in a Distributed Environment Using eXtensible Markup Language (XML). In: 2001 ACM Symposium on Applied Computing (SAC 2001), Las Vegas, NV, pp. 405–411 (2001)
40. W3C, SOAP Specifications, W3C Recommendation, 2nd edn. (April 27, 2007)
41. W3C, Web Services Description Language (WSDL) 1.1, W3C Note (March 15, 2001)

Computations and Interaction

Jos C.M. Baeten[1,2], Bas Luttik[2,3], and Paul van Tilburg[2]

[1] Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2] Division of Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[3] Department of Computer Science, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{j.c.m.baeten,s.p.luttik,p.j.a.v.tilburg}@tue.nl

Abstract. We enhance the notion of a computation of the classical theory of computing with the notion of interaction. In this way, we enhance a Turing machine as a model of computation to a Reactive Turing Machine that is an abstract model of a computer as it is used nowadays, always interacting with the user and the world.

1 Introduction

What is a computation? This is a central question in the theory of computing, dating back to 1936 [17]. The classical answer is that a computation is given by a Turing machine, with the input given on its tape at the beginning, after which a sequence of steps takes place, leaving the output on the tape at the end. A computable function is a function of which the transformation of input to output can be computed by a Turing machine.

A Turing machine can serve in this way as a basic model of a computation, but cannot serve as a basic model of a computer. Well, it could up to the advent of the terminal in the 1970s. Before that, input was given as a stack of punch cards at the start, and output of a computation appeared as a printout later. The terminal made direct interaction with the computer possible. Nowadays, a computer is interacting continuously, with the user at the click of a mouse or with many other computers all over the world through the Internet.

An execution of a computer is thus not just a series of steps of a computation, but also involves interaction. It cannot be modeled as a function, and has inherent nondeterminism. In this paper, we make the notion of an execution precise, and compare this to the notion of a computation. To illustrate the difference between a computation and an execution, we can say that a Turing machine cannot fly an airplane, but a computer can. An automatic pilot cannot know all weather conditions en route beforehand, but can react to changing conditions real-time.

Computability theory is firmly grounded in automata theory and formal language theory. It progresses from the study of finite automata to pushdown automata and Turing machines. Of these different classes of automata, it studies the languages, the sets of strings, induced by them. We can view a language as an equivalence class of automata (under language equivalence).

The notion of interaction has been studied extensively in concurrency theory and process theory, see e.g. [13]. It embodies a powerful parallel composition operator that is used to compose systems in parallel, including their interaction. The semantics of concurrency theory is mostly given in terms of transition systems, that are almost like automata. However, there are important differences.

First of all, a notion of final state, of termination, is often missing in concurrency theory. The idea is that concurrency theory often deals with so-called *reactive systems*, which need not terminate but are always on, reacting to stimuli from the environment. As a result, termination is often neglected in concurrency theory, but is nevertheless an important ingredient, as shown and fully worked out in [2]. Using this presentation of concurrency theory as a starting point, we obtain a full correspondence with automata theory: a finite transition system is exactly a finite automaton. On the other hand, we stress that we fully incorporate the reactive systems approach of concurrency theory: non-terminating behaviour is also relevant behaviour, that is taken into account.

A second difference between automata theory and concurrency theory is that transition systems need not be finite. Still, studying the subclass of finite transition systems yields useful insights for the extension to pushdown automata and Turing machines.

The third and main difference between automata theory and concurrency theory is that language equivalence is too coarse to capture a notion of interaction. Looking at an automaton as a language acceptor, acceptance of a string represents a particular computation of the automaton, and the language is the set of all its computations. The language-theoretic interpretation abstracts from the moments of choice within an automaton. For instance, it does not distinguish between, on the one hand, the automaton that first accepts an a and subsequently chooses between accepting a b or a c, and, on the other hand, the automaton that starts with a choice between accepting ab and accepting ac. As a consequence, the language-theoretic interpretation is only suitable under the assumption that an automaton is a stand-alone computational device; it is unsuitable if some form interaction of the automaton with its environment (user, other automata running in parallel, etc.) may influence the course of computation.

Therefore, other notions of equivalence are studied in concurrency theory, capturing more of the branching structure of an automaton. Prominent among these is bisimulation equivalence [15]. When silent steps are taken into account, the preferred variant is *branching bisimilarity*, arguably preserving all relevant moments of choice in a system [11].

In this paper we study the notion of a computation, taking interaction into account. We define, next to the notion of a computable function, the notion of an executable process. An executable process is a behaviour that can be exhibited by a computer (interacting with its environment). An executable process is a branching bisimulation equivalence class of transition systems defined by a Reactive Turing Machine. A Reactive Turing Machine is an adaptation of the classical

Turing Machine that can properly deal with ubiquitous interaction. Leading up to the definition of the Reactive Turing Machine, we reconsider some of the standard results from automata theory when automata are considered modulo *branching bisimilarity* instead of language equivalence.

In Section 3 we consider *finite-state processes*, defined as branching bisimulation equivalence classes of finite labeled transition systems that are finite automata. The section illustrates the correspondence between finite automata and linear recursive specifications that can be thought of as the process-theoretic counterpart of regular grammars.

In Section 4 we consider *pushdown processes*, defined as branching bisimulation equivalence classes of labeled transition systems associated with pushdown automata. We investigate the correspondence between pushdown processes and processes definable by sequential recursive specifications, which can be thought of as the process-theoretic counterpart of context-free grammars.

In Section 5 we define *executable processes*, defined as branching bisimulation equivalence classes of labeled transition systems associated with Reactive Turing Machines. We highlight the relationship of computable functions and executable processes, laying the foundations of executability theory alongside computability theory.

2 Process Theory

In this section we briefly recap the basic definitions of the process algebra TCP$^*_\tau$ (Theory of Communicating Processes with silent step and iteration). This process algebra has a rich syntax, allowing to express all key ingredients of concurrency theory, including termination that enables a full correspondence with automata theory. It also has a rich theory, fully worked out in [2].

Syntax. We presuppose a finite *action alphabet* $\mathcal{A}$, and a countably infinite set of *names* $\mathcal{N}$. The actions in $\mathcal{A}$ denote the basic events that a process may perform. We furthermore presuppose a finite *data alphabet* $\mathcal{D}$, a finite set $\mathcal{C}$ of *channels*, and assume that $\mathcal{A}$ includes special actions $c?d$, $c!d$, $c\,!\!?\,d$ ($d \in \mathcal{D}$, $c \in \mathcal{C}$), which, intuitively, denote the event that datum d is received, sent, or communicated along channel c.

Let $\mathcal{N}'$ be a finite subset of $\mathcal{N}$. The set of *process expressions* $\mathcal{P}$ over $\mathcal{A}$ and $\mathcal{N}'$ is generated by the following grammar:

$$p ::= \mathbf{0} \mid \mathbf{1} \mid a.p \mid \tau.p \mid p \cdot p \mid p^* \mid p + p \mid p \parallel p \mid \partial_c(p) \mid \tau_c(p) \mid N \qquad (a \in \mathcal{A}, N \in \mathcal{N}', c \in \mathcal{C}).$$

Let us briefly comment on the operators in this syntax. The constant $\mathbf{0}$ denotes inaction or *deadlock*, the unsuccessfully terminated process. It can be thought of as the automaton with one initial state that is not final and no transitions. The constant $\mathbf{1}$ denotes the successfully terminated process. It can be thought of as the automaton with one initial state that is final, without transitions. For each action $a \in \mathcal{A}$ there is a unary operator $a.$ denoting action prefix; the process denoted by $a.p$ can do an a-transition to the process denoted by p. The τ-transitions of a process will, in the semantics below, be treated as unobservable, and as such they are the process-theoretic counterparts of the so-called λ- or ϵ-transitions in the theory of automata and formal languages. We write $\mathcal{A}_\tau$ for $\mathcal{A} \cup \{\tau\}$. The binary operator $\cdot$ denotes *sequential composition.* The unary operator * is iteration or *Kleene star.* The binary operator $+$ denotes *alternative composition* or *choice.* The binary operator $\parallel$ denotes *parallel composition*; actions of both arguments are interleaved, and in addition a communication $c?\!\!\!\!!\,d$ of a datum d on channel c can take place if one argument can do an input action $c?d$ that matches an output action $c!d$ of the other component. The unary operator $\partial_c(p)$ encapsulates the process p in such a way that all input actions $c?d$ and output actions $c!d$ are blocked (for all data) so that communication is enforced. Finally, the unary operator $\tau_c(p)$ denotes abstraction from communication over channel c in p by renaming all communications $c?\!\!\!\!!\,d$ to τ-transitions.

Let $\mathcal{N}'$ be a finite subset of $\mathcal{N}$, used to define processes by means of (recursive) equations. A *recursive specification* E over $\mathcal{N}'$ is a set of equations of the form $N \overset{\text{def}}{=} p$ with as left-hand side a name N and as right-hand side a process expression p. It is required that a recursive specification E contains, for every $N \in \mathcal{N}'$, precisely one equation with N as left-hand side.

One way to formalize the operational intuitions we have for the syntactic constructions of TCP^*_τ, is to associate with every process expression a labeled transition system.

Definition 1 (Labeled Transition System). *A* labeled transition system L *is defined as a four-tuple* $(\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ *where:*

1. $\mathcal{S}$ *is a set of states,*
2. $\rightarrow \subseteq \mathcal{S} \times \mathcal{A}_\tau \times \mathcal{S}$ *is an $\mathcal{A}_\tau$-labeled* transition relation *on* $\mathcal{S}$,
3. $\uparrow \in \mathcal{S}$ *is the* initial *state,*
4. $\downarrow \subseteq \mathcal{S}$ *is the set of* final *states.*

If $(s, a, t) \in \rightarrow$, *we write* $s \xrightarrow{a} t$. *If* s *is a final state, i.e.,* $s \in \downarrow$, *we write* $s\downarrow$.

We see that a labeled transition system with a finite set of states is exactly a finite (nondeterministic) automaton.

We use Structural Operational Semantics [16] to associate a transition relation with process expressions: we let $\rightarrow$ be the $\mathcal{A}_\tau$-labeled transition relation induced on the set of process expressions $\mathcal{P}$ by the operational rules in Table 1. Note that the operational rules presuppose a recursive specification E.

Let $\rightarrow$ be an $\mathcal{A}_\tau$-labeled transition relation on a set $\mathcal{S}$ of states. For $s, s' \in \mathcal{S}$ and $w \in \mathcal{A}^*$ we write $s \overset{w}{\twoheadrightarrow} s'$ if there exist states $s_0, \ldots, s_n \in \mathcal{S}$ and actions $a_1, \ldots, a_n \in \mathcal{A}_\tau$ such that $s = s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n = s'$ and w is obtained from $a_1 \cdots a_n$ by omitting all occurrences of τ. ε denotes the empty word. We say a state $t \in \mathcal{S}$ is *reachable* from a state $s \in \mathcal{S}$ if there exists $w \in \mathcal{A}^*$ such that $s \overset{w}{\twoheadrightarrow} t$.

Table 1. Operational rules for TCP_τ^* and a recursive specification E (a ranges over $\mathcal{A}_\tau$, d ranges over $\mathcal{D}$, and c ranges over $\mathcal{C}$)

$$\frac{}{\mathbf{1}\downarrow} \qquad \frac{}{p^*\downarrow} \qquad \frac{}{a.p \xrightarrow{a} p}$$

$$\frac{p \xrightarrow{a} p'}{(p+q) \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{(p+q) \xrightarrow{a} q'} \qquad \frac{p\downarrow}{(p+q)\downarrow} \qquad \frac{q\downarrow}{(p+q)\downarrow}$$

$$\frac{p \xrightarrow{a} p'}{p \cdot q \xrightarrow{a} p' \cdot q} \qquad \frac{p\downarrow \quad q \xrightarrow{a} q'}{p \cdot q \xrightarrow{a} q'} \qquad \frac{p\downarrow \quad q\downarrow}{p \cdot q\downarrow} \qquad \frac{p \xrightarrow{a} p'}{p^* \xrightarrow{a} p' \cdot p^*}$$

$$\frac{p \xrightarrow{a} p'}{p \parallel q \xrightarrow{a} p' \parallel q} \qquad \frac{q \xrightarrow{a} q'}{p \parallel q \xrightarrow{a} p \parallel q'} \qquad \frac{p\downarrow \quad q\downarrow}{p \parallel q\downarrow}$$

$$\frac{p \xrightarrow{c!d} p' \quad q \xrightarrow{c?d} q'}{p \parallel q \xrightarrow{c?\!\!\!!\,d} p' \parallel q'} \qquad \frac{p \xrightarrow{c?d} p' \quad q \xrightarrow{c!d} q'}{p \parallel q \xrightarrow{c?\!\!\!!\,d} p' \parallel q'}$$

$$\frac{p \xrightarrow{a} p' \quad a \neq c?d, c!d}{\partial_c(p) \xrightarrow{a} \partial_c(p')} \qquad \frac{p\downarrow}{\partial_c(p)\downarrow}$$

$$\frac{p \xrightarrow{c?\!\!\!!\,d} p'}{\tau_c(p) \xrightarrow{\tau} \tau_c(p')} \qquad \frac{p \xrightarrow{a} p' \quad a \neq c?\!\!\!!\,d}{\tau_c(p) \xrightarrow{a} \tau_c(p')} \qquad \frac{p\downarrow}{\tau_c(p)\downarrow}$$

$$\frac{p \xrightarrow{a} p' \quad (N \stackrel{\text{def}}{=} p) \in E}{N \xrightarrow{a} p'} \qquad \frac{p\downarrow \quad (N \stackrel{\text{def}}{=} p) \in E}{N\downarrow}$$

Definition 2. *Let E be a recursive specification and let p be a process expression. We define the labeled transition system $\mathcal{T}_E(p) = (\mathcal{S}_p, \rightarrow_p, \uparrow_p, \downarrow_p)$ associated with p and E as follows:*

1. *the set of states $\mathcal{S}_p$ consists of all process expressions reachable from p;*
2. *the transition relation $\rightarrow_p$ is the restriction to $\mathcal{S}_p$ of the transition relation $\rightarrow$ defined on all process expressions by the operational rules in Table 1, i.e., $\rightarrow_p = \rightarrow \cap (\mathcal{S}_p \times \mathcal{A}_\tau \times \mathcal{S}_p)$.*
3. *the process expression p is the initial state, i.e. $\uparrow_p = p$; and*
4. *the set of final states consists of all process expressions $q \in \mathcal{S}_p$ such that $q\downarrow$, i.e., $\downarrow_p = \downarrow \cap \mathcal{S}_p$.*

If we start out from a process expression not containing a name, then the transition system defined by this construction is finite and so is a finite automaton.

Given the set of (possibly infinite) labeled transition systems, we can divide out different equivalence relations on this set. Dividing out language equivalence throws away too much information, as the moments where choices are made are totally lost, and behavior that does not lead to a final state is ignored.

An equivalence relation that keeps all relevant information, and has many good properties, is branching bisimulation as proposed by van Glabbeek and Weijland [11]. For motivations to use branching bisimulation as the preferred notion of equivalence, see [9].

Let $\rightarrow$ be an $\mathcal{A}_\tau$-labeled transition relation, and let $a \in \mathcal{A}_\tau$; we write $s \xrightarrow{(a)} t$ if $s \xrightarrow{a} t$ or $a = \tau$ and $s = t$.

Definition 3 (Branching bisimilarity). *Let $L_1 = (\mathcal{S}_1, \rightarrow_1, \uparrow_1, \downarrow_1)$ and $L_2 = (\mathcal{S}_2, \rightarrow_2, \uparrow_2, \downarrow_2)$ be labeled transition systems. A* branching bisimulation *from L_1 to L_2 is a binary relation $\mathcal{R} \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ such that $\uparrow_1 \mathcal{R} \uparrow_2$ and, for all states s_1 and s_2, $s_1 \mathcal{R} s_2$ implies*

1. *if $s_1 \xrightarrow{a}_1 s_1'$, then there exist $s_2', s_2'' \in \mathcal{S}_2$ such that $s_2 \overset{\varepsilon}{\twoheadrightarrow}_2 s_2'' \xrightarrow{(a)}_2 s_2'$, $s_1 \mathcal{R} s_2''$ and $s_1' \mathcal{R} s_2'$;*
2. *if $s_2 \xrightarrow{a}_2 s_2'$, then there exist $s_1', s_1'' \in \mathcal{S}_1$ such that $s_1 \overset{\varepsilon}{\twoheadrightarrow}_1 s_1'' \xrightarrow{(a)}_1 s_1'$, $s_1'' \mathcal{R} s_2$ and $s_1' \mathcal{R} s_2'$;*
3. *if $s_1\downarrow_1$, then there exists s_2' such that $s_2 \overset{\varepsilon}{\twoheadrightarrow}_2 s_2'$ and $s_2'\downarrow_2$; and*
4. *if $s_2\downarrow_2$, then there exists s_1' such that $s_1 \overset{\varepsilon}{\twoheadrightarrow}_1 s_1'$ and $s_1'\downarrow_1$.*

The labeled transition systems L_1 and L_2 are branching bisimilar *(notation: $L_1 \leftrightarrows_b L_2$) if there exists a branching bisimulation from L_1 to L_2.*

Branching bisimilarity is an equivalence relation on labeled transition systems [8]. A branching bisimulation from a transition system to itself is called a branching bisimulation *on* this transition system. Each transition system has a maximal branching bisimulation, identifying as many states as possible, found as the union of all possible branching bisimulations. Dividing out this maximal branching bisimulation, we get the quotient of the transition system w.r.t. the maximal branching bisimulation. We define the *branching degree* of a state as the cardinality of the set of outgoing edges of its equivalence class in the maximal branching bisimulation.

A transition system has *finite branching* if all states have a finite branching degree. We say a transition system has *bounded branching* if there exists a natural number $n \geq 0$ such that every state has a branching degree of at most n. Branching bisimulations respect branching degrees.

3 Regular Processes

A computer with a fixed-size, finite memory is just a finite control. This can be modeled by a finite automaton. Automata theory starts with the notion of a finite automaton. As nondeterminism is relevant and basic in concurrency theory, we look at a nondeterministic finite automaton. As a nondeterministic finite automaton is exactly a finite labeled transition system, we refer to Definition 1 for the definition.

Two examples of finite automata are given in Figure 1.

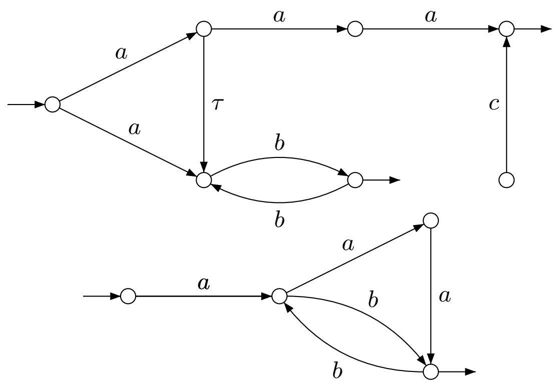


Fig. 1. Two examples of finite automata

Definition 4 (Deterministic finite automaton). *A finite automaton $M = (\mathcal{S}, \mathcal{A}, \rightarrow, \uparrow, \downarrow)$ is* deterministic *if, for all states $s, t_1, t_2 \in \mathcal{S}$ and for all actions $a \in \mathcal{A}'$, $s \overset{\varepsilon}{\twoheadrightarrow} \xrightarrow{a} t_1$ and $s \overset{\varepsilon}{\twoheadrightarrow} \xrightarrow{a} t_2$ implies $t_1 = t_2$.*

In the theory of automata and formal languages, it is usually also required in the definition of deterministic that the transition relation is *total* in the sense that for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}'$ there exists $t \in \mathcal{S}$ such that $s \xrightarrow{a} t$. The extra requirement is clearly only sensible in the language interpretation of automata; we shall not be concerned with it here.

The upper automaton in Figure 1 is nondeterministic and has an unreachable c-transition. The lower automaton is deterministic and does not have unreachable transitions; it is not total.

In the theory of automata and formal languages, finite automata are considered as language acceptors.

Definition 5 (Language equivalence). *The* language $\mathcal{L}(L)$ *accepted by a labeled transition system $L = (\mathcal{S}, \rightarrow, \uparrow, \downarrow)$ is defined as*

$$\mathcal{L}(L) = \{w \in \mathcal{A}^* \mid \exists s \in \downarrow \text{ such that } \uparrow \overset{w}{\twoheadrightarrow} s\}.$$

Labeled transition systems L_1 and L_2 are language equivalent *(notation: $L_1 \equiv L_2$) if $\mathcal{L}(L_1) = \mathcal{L}(L_2)$.*

The language of both automata in Figure 1 is $\{aaa\} \cup \{ab^{2n-1} \mid n \geq 1\}$; the automata are language equivalent.

A language $L \subseteq \mathcal{A}^*$ accepted by a finite automaton is called a *regular language*. A *regular process* is a branching bisimilarity class of labeled transition systems that contains a finite automaton.

In automata theory, every silent step τ and all nondeterminism can be removed from a finite automaton. These results are no longer valid when we consider finite automata modulo branching bisimulation. Not every regular process has a representation as a finite automaton without τ-transitions, and not every regular process has a representation as a deterministic finite automaton. In fact, it can be proved that there does not exist a finite automaton without τ-transitions that is branching bisimilar with the upper finite automaton in Figure 1. Nor does there exist a deterministic finite automaton branching bisimilar with the upper finite automaton in Figure 1.

Regular expressions. A *regular expression* is a process expression using only the first 7 items in the definition of process syntax above, so does not contain parallel composition or recursion. Not every regular process is given by a regular expression, see [3]. We show a simple example in Figure 2 of a finite transition system that is not bisimilar to any transition system that can be associated with a regular expression.

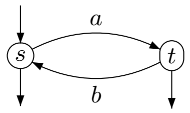


Fig. 2. Not bisimilar to a regular expression

However, if we can also use parallel composition and encapsulation, then we can find an expression for every finite automaton, see [7]. Abstraction and recursion are not needed for this result. We can illustrate this with the finite automaton in Figure 2, but need to replace the label a by $st?a$ and label b by $ts?b$. Then, we can define the following expressions for states s, t:

$$s = (ts?b.(st!a.\mathbf{1} + \mathbf{1}))^*, \qquad t = (st?a.(ts!b.\mathbf{1} + \mathbf{1}))^*.$$

The expressions show the possibilities to enter a state, followed by the possibilities to leave a state, and then iterate. Then, we compose the expressions of the states in a parallel composition:

$$\partial_{st,ts}(((st!a.\mathbf{1} + \mathbf{1}) \cdot s) \parallel \mathbf{1} \cdot t).$$

Using the operational rules on this resulting expression gives again the finite automaton in Figure 2.

Regular grammars. In the theory of automata and formal languages, the notion of *grammar* is used as a syntactic mechanism to describe languages. The corresponding mechanism in concurrency theory is the notion of recursive specification.

If we use only the syntax elements $\mathbf{0}$, $\mathbf{1}$, N ($N \in \mathcal{N}'$), $a.\_$ ($a \in \mathcal{A}_\tau$) and $\_+\_$ of the definition above, then we get so-called *linear* recursive specifications. Thus, we do not use sequential composition, parallel composition, encapsulation and abstraction.

We have the result that every linear recursive specification by means of the operational rules defined generates a finite automaton, but also conversely, every finite automaton can be specified, up to isomorphism, by a linear recursive specification. We illustrate the construction with an example.

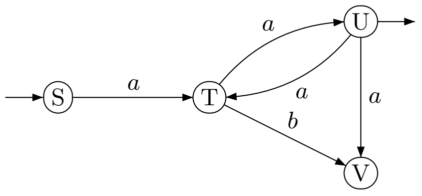


Fig. 3. Example automaton

Consider the automaton depicted in Figure 3. Note that we have labeled each state of the automaton with a unique name; these will be the names of a recursive specification E. We will define each of these names with an equation, in such a way that the labeled transition system $\mathcal{T}_E(S)$ generated by the operational semantics in Table 1 is isomorphic (so certainly branching bisimilar) with the automaton in Figure 3.

The recursive specification for the finite automaton in Figure 3 is:

$$S \overset{\text{def}}{=} a.T, \qquad T \overset{\text{def}}{=} a.U + b.V, \qquad U \overset{\text{def}}{=} a.V + \mathbf{1}, \qquad V \overset{\text{def}}{=} \mathbf{0}.$$

This result can be viewed as the process-theoretic counterpart of the result from the theory of automata and formal languages that states that every language accepted by a finite automaton is generated by a so-called *right-linear* grammar. There is no reasonable process-theoretic counterpart of the similar result in the theory of automata and formal languages that every language accepted by a finite automaton is generated by a *left-linear* grammar. If we use *action postfix* instead of action prefix, then on the one hand not every finite automaton can be specified, and on the other hand, by means of a simple recursive equation we can specify an infinite transition system (see [4]).

We conclude that the classes of processes defined by right-linear and left-linear grammars do not coincide.

4 Pushdown and Context-Free Processes

As an intermediate between the notions of finite automaton and Turing machine, the theory of automata and formal languages treats the notion of pushdown automaton, which is a finite automaton with a stack as memory. Several definitions of the notion appear in the literature, which are all equivalent in the sense that they accept the same languages.

Definition 6 (Pushdown automaton). *A* pushdown automaton *M is defined as a six-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ where:*

1. *$\mathcal{S}$ a finite set of states,*
2. *$\mathcal{A}$ is a finite action alphabet,*
3. *$\mathcal{D}$ is a finite data alphabet,*

4. *$\rightarrow \subseteq \mathcal{S} \times \mathcal{A}_\tau \times (\mathcal{D} \cup \{\varepsilon\}) \times \mathcal{D}^* \times \mathcal{S}$ is a $\mathcal{A}_\tau \times (\mathcal{D} \cup \{\varepsilon\}) \times \mathcal{D}^*$-labeled transition relation on $\mathcal{S}$,*
5. *$\uparrow \in \mathcal{S}$ is the initial state, and*
6. *$\downarrow \subseteq \mathcal{S}$ is the set of final states.*

If $(s, a, d, \delta, t) \in \rightarrow$, we write $s \xrightarrow{a[d/\delta]} t$.

The pair of a state together with particular stack contents will be referred to as the *configuration* of a pushdown automaton. Intuitively, a transition $s \xrightarrow{a[d/\delta]} t$ (with $a \in \mathcal{A}$) means that the automaton, when it is in a configuration consisting of a state s and a stack with the datum d on top, can consume input symbol a, replace d by the string δ and move to state t. Likewise, writing $s \xrightarrow{a[\varepsilon/\delta]} t$ means that the automaton, when it is in state s and the stack is empty, can consume input symbol a, put the string δ on the stack, and move to state t. Transitions of the form $s \xrightarrow{\tau[d/\delta]} t$ or $s \xrightarrow{\tau[\varepsilon/\delta]} t$ do not entail the consumption of an input symbol, but just modify the stack contents.

When considering a pushdown automaton as a language acceptor, it is generally assumed that it starts in its initial state with an empty stack. A computation consists of repeatedly consuming input symbols (or just modifying stack contents without consuming input symbols). When it comes to determining whether or not to accept an input string there are two approaches: "acceptance by final state" (FS) and "acceptance by empty stack" (ES). The first approach accepts a string if the pushdown automaton can move to a configuration with a final state by consuming the string, ignoring the contents of the stack in this configuration. The second approach accepts the string if the pushdown automaton can move to a configuration with an empty stack, ignoring whether the state of this configuration is final or not. These approaches are equivalent from a language-theoretic point of view, but not from a process-theoretic point of view. We also have a third approach in which a configuration is terminating if it consists of a terminating state *and* an empty stack (FSES). We note that, from a process-theoretic point of view, the ES and FSES approaches lead to the same notion of pushdown process, whereas the FS approach leads to a different notion. We established in [4] that for FS, the connection with context-free grammars is more difficult to make (see further on). That is why we adopt the FSES approach in the sequel.

Definition 7. *Let $M = (\mathcal{S}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ be a pushdown automaton. The labeled transition system $\mathcal{T}(M)$ associated with M is defined as follows:*

1. *the set of states of $\mathcal{T}(M)$ is $\mathcal{S} \times \mathcal{D}^*$;*
2. *the transition relation of $\mathcal{T}(M)$ satisfies*
   (a) $(s, d\zeta) \xrightarrow{a} (t, \delta\zeta)$ iff $s \xrightarrow{a[d/\delta]} t$ for all $s, t \in \mathcal{S}$, $a \in \mathcal{A}_\tau$, $d \in \mathcal{D}$, $\delta, \zeta \in \mathcal{D}^*$, and
   (b) $(s, \varepsilon) \xrightarrow{a} (t, \delta)$ iff $s \xrightarrow{a[\varepsilon/\delta]} t$;
3. *the initial state of $\mathcal{T}(M)$ is $(\uparrow, \varepsilon)$; and*
4. *the set of final states is $\{(s, \varepsilon) \mid s\downarrow\}$.*

This definition now gives us the notions of pushdown language and pushdown process: a *pushdown language* is the language of the transition system associated with a pushdown automaton, and a *pushdown process* is a branching bisimilarity class of labeled transition systems containing a labeled transition system associated with a pushdown automaton.

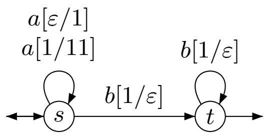


Fig. 4. Example pushdown automaton

As an example, the pushdown automaton in Figure 4 defines the infinite transition system in Figure 5, that accepts the language $\{a^n b^n \mid n \geq 0\}$.

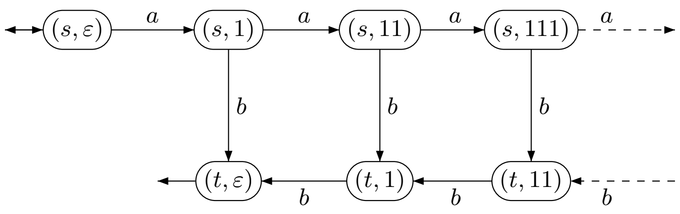


Fig. 5. A pushdown process

Only push and pop transitions. It is not difficult to see that limiting the set of transitions to push and pop transitions only in the definition of pushdown automaton yields the same notion of pushdown process. Here, a *push* transition is a transition with label $a[\varepsilon/d]$ or $a[d/ed]$, and a *pop* transition is a transition with label $a[d/\varepsilon]$.

Context-free grammars. We shall now consider the process-theoretic version of the standard result in the theory of automata and formal languages that the set of pushdown languages coincides with the set of languages generated by context-free grammars. As the process-theoretic counterparts of context-free grammars we shall consider so-called *sequential* recursive specifications in which only the constructions $\mathbf{0}$, $\mathbf{1}$, N $(N \in \mathcal{N}')$, $a.\_$ $(a \in \mathcal{A}_\tau)$, $\_ \cdot \_$ and $\_ + \_$ occur, so adding sequential composition to linear recursive specifications.

Sequential recursive specifications can be used to specify pushdown processes. To give an example, the process expression X defined in the sequential recursive specification

$$X \overset{\text{def}}{=} \mathbf{1} + a.X \cdot b.\mathbf{1}$$

specifies the labeled transition system in Figure 5, which is associated with the pushdown automaton in Figure 4.

If we would consider pushdown automata with termination just by final state (irrespective of the contents of the stack), then the transition system of this pushdown automaton would have every state final, and this cannot be realised by a sequential recursive specification [4].

The notion of a sequential recursive specification naturally corresponds with with the notion of context-free grammar: for every pushdown automaton there exists a sequential recursive specification such that their transition systems are language equivalent, and, vice versa, for every sequential recursive specification there exists a pushdown automaton such that their transition systems are language equivalent.

A similar result with language equivalence replaced by branching bisimilarity does not hold. In fact, we shall see that there are pushdown processes that are not recursively definable by a sequential recursive specification, and that there are also sequential recursive specifications that define non-pushdown processes. We shall present a restriction on pushdown automata and a restriction on sequential recursive specifications that enable us to retrieve the desired equivalence: we proved in [4] that the set of so-called *popchoice-free* pushdown processes corresponds with the set of processes definable by a *transparency-restricted* sequential recursive specification without *head recursion*. Our result is not optimal: we give an example of a pushdown process that is not popchoice-free, but is definable by a sequential recursive specification.

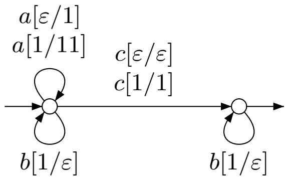


Fig. 6. Pushdown automaton that is not popchoice-free

Consider the pushdown automaton in Figure 6, which generates the transition system shown in Figure 7. In [14], Moller proved that this transition system cannot be defined with a BPA recursive specification, where BPA is the restriction of sequential recursive specifications by omitting the τ-prefix and the constant **0** and by disallowing **1** to occur as a summand in a nontrivial alternative composition. His proof can be modified to show that the transition system is not definable with a sequential recursive specification either. We conclude that not every pushdown process is definable with a sequential recursive specification.

Note that a push of a 1 onto the stack in the initial state of the pushdown automaton in Figure 6 can (on the way to termination) be popped again in the initial state or in the final state: the choice of where the pop will take place cannot be made at the time of the push. In other words, in the pushdown automaton in Figure 6 pop transitions may induce a choice in the associated transition system; we refer to such choice through a pop transition as a *popchoice*. By disallowing popchoice we define a class of pushdown processes that are definable with a sequential recursive specification.

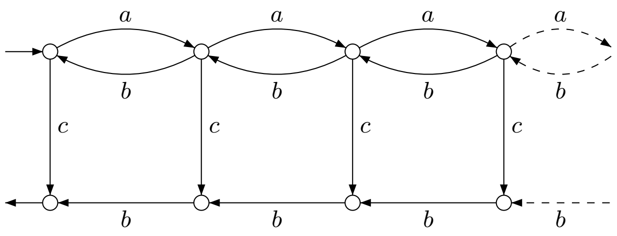


Fig. 7. Transition system of automaton of Figure 6

Suppose we have a pushdown automaton that uses only push and pop transitions. A d-pop transition is a transition with label $a[d/\varepsilon]$. We say the pushdown automaton is *popchoice-free* iff whenever there are two d-pop transitions, they lead to the same state. A pushdown process is *popchoice-free* if it contains a labeled transition system associated with a popchoice-free pushdown automaton.

The definition of a pushdown automaton uses a stack as memory. The stack itself can be modeled as a pushdown process, in fact (as we will see shortly) it is the prototypical pushdown process. Given a finite data set $\mathcal{D}$, the stack has an input channel i over which it can receive elements of $\mathcal{D}$ and an output channel o over which it can send elements of $\mathcal{D}$. The stack process is given by a pushdown automaton with one state $\uparrow$ (which is both initial and final) and transitions $\uparrow \xrightarrow{i?d[\varepsilon/d]} \uparrow$, $\uparrow \xrightarrow{i?d[e/de]} \uparrow$, and $\uparrow \xrightarrow{o!d[d/\varepsilon]} \uparrow$ for all $d, e \in \mathcal{D}$. As this pushdown automaton has only one state, it is popchoice-free. The following recursive specification E_S defines a stack over data set $\mathcal{D}$:

$$S \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} i?d.S \cdot o!d.S.$$

The stack process can be used to make the interaction between control and memory in a pushdown automaton explicit [5]. This is illustrated by the following theorem, stating that every pushdown process is equal to a regular process interacting with a stack.

Theorem 1. *For every pushdown automaton M there exists a regular process expression p and a linear recursive specification E, and for every regular process expression p and linear recursive specification there exists a pushdown automaton M such that*

$$\mathcal{T}(M) \leftrightarroweq_b \mathcal{T}_{E \cup E_S}(\tau_{i,o}(\partial_{i,o}(p \parallel S))).$$

In automata theory, context-free grammars are often reduced to a normal form, by which they become more tractable. Some of these transformations can be performed on sequential recursive specifications as well, preserving branching bisimulation. Others cannot. Useful in our case will be the restricted Greibach normal form, see [12].

Every sequential recursive specification can be brought into *Process Greibach normal form*, that is, satisfying the requirement that every right-hand side of the equation of N only has summands that are $\mathbf{1}$ or of the forms $a.\mathbf{1}$, $a.X$, $a.X \cdot Y$ or $N \cdot X$ for certain names X, Y. Compared to the case of automata theory, we cannot remove $\mathbf{1}$ summands and we cannot remove *head recursion* (where a name has a summand that is a sequential composition of itself with another or the same name).

A convenient property of recursive specification in Process Greibach normal form is that every reachable state in the labeled transition system associated with a name N in such a recursive specification will be denoted by a sequential composition of names (see, e.g., the labeled transition system in Figure 8).

Let p be a process expression in the context of a sequential recursive specification E. In case there is no head recursion, the associated labeled transition system $\mathcal{T}_E(p)$ has finite branching (see, e.g., [2] for a proof). Using head recursion, this is possible, as the following example shows.

$$X \overset{\text{def}}{=} \mathbf{1} + X \cdot Y, \qquad\qquad Y \overset{\text{def}}{=} a.\mathbf{1}$$

Still, restricting to sequential recursive specifications in Process Greibach normal form is not sufficient to get the desired correspondence between processes definable by sequential recursive specifications and processes definable as a popchoice-free pushdown automaton. Consider the following recursive specification, which is in Process Greibach normal form and has no head recursion:

$$X \overset{\text{def}}{=} a.X \cdot Y + b.\mathbf{1}, \qquad Y \overset{\text{def}}{=} \mathbf{1} + c.\mathbf{1}.$$

The labeled transition system associated with X, which is depicted in Figure 8, has finite but unbounded branching. We claim this cannot be a pushdown process.

Note that the unbounded branching is due to the $\mathbf{1}$-summand in the equation of Y by which $Y^n \xrightarrow{c} Y^m$ for all $m < n$. A name N in a recursive specification is called *transparent* if its equation has a $\mathbf{1}$-summand; otherwise it is called *opaque*. To exclude recursive specifications generating labeled transition systems with unbounded branching, we will require that transparent names may only occur as the *last* element of reachable sequential compositions of names.

Thus, we call a sequential recursive specification in Process Greibach normal form *transparency-restricted* if for all (generalized) sequential compositions of names reachable from a name in the specification it holds that all but the last name is opaque.

As an example, note that the specification of the stack over $\mathcal{D}$ defined above is not transparency restricted, because it is not in Process Greibach normal form. But the same process can be defined with a transparency-restricted recursive specification, without head recursion: it suffices to add, for all $d \in \mathcal{D}$, a name T_d to replace $S \cdot o!d.\mathbf{1}$. Thus we obtain the following transparency-restricted specification of the stack over $\mathcal{D}$:

$$S \overset{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}'} i?d.T_d \cdot S, \qquad T_d \overset{\text{def}}{=} o!d.\mathbf{1} + \sum_{e \in \mathcal{D}'} i?e.T_e \cdot T_d.$$

It can be seen that the labeled transition system associated with a name in a transparency-restricted specification has either bounded or infinite branching.

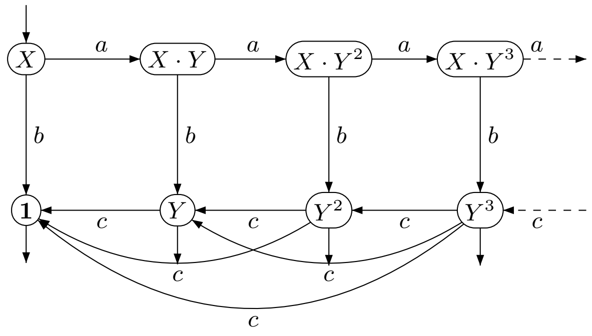


Fig. 8. Process with unbounded branching

In the case without head recursion, the branching degree of a state denoted by a reachable sequential composition of names is equal to the branching degree of its first name, and the branching degree of a name is bounded by the number of summands of the right-hand side of its defining equation. In the case with head recursion, when name N has a summand $N \cdot P$ and there is a transition $N \xrightarrow{a} \xi$ for a sequence of names ξ, then there are steps $N \xrightarrow{a} \xi \cdot M^n$ for any n-fold sequence of names M, making the branching infinite.

For investigations under what circumstances we can extend the set of pushdown processes to incorporate processes with finite but unbounded branching, see [5]. In this paper a (partially) forgetful stack is used to deal with transparent variables on the stack. However, if we allow for τ-transitions in the recursive specifications, we can use the stack as is presented above. Note also that the paper does not require the recursive specifications to be transparency-restricted, but this comes at the cost of using a weaker equivalence (namely contrasimulation [10] instead of branching bisimulation) in some cases.

We are now in a position to establish a process-theoretic counterpart of the correspondence between pushdown automata and context-free grammars.

Theorem 2. *A process is a popchoice-free pushdown process if, and only if, it is definable by a transparency-restricted recursive specification without head recursion.*

This theorem was proven in [4].

Consider the pushdown automaton shown in Figure 4. This pushdown automaton is popchoice-free, since both 1-pop transitions lead to the same state. The method described in the proof of Theorem 2 can now be used to yield a sequential recursive specification. After simplification, this specification again reduces to the specification we had before, $X = \mathbf{1} + a.X \cdot b.\mathbf{1}$.

Thus, we have established a correspondence between a popchoice-free pushdown processes on the one hand, and transparency-restricted recursive specification on the other hand, thereby casting the classical result of the equivalence of pushdown automata and context-free grammars in terms of processes and bisimulation.

To show that this result is not optimal, consider the following sequential recursive specification:

$$X \overset{\text{def}}{=} a.1 + X \cdot Y, \qquad Y \overset{\text{def}}{=} b.1$$

This specification is in Process Greibach normal form, and is transparency-restricted, as all variables are opaque. The resulting transition system is infinitely branching. However, inserting additional τ-steps, we can see it is branching bisimilar to the transition system in Figure 9.

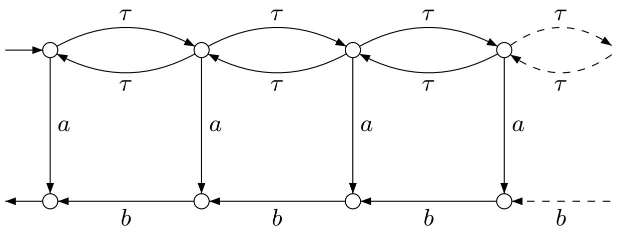


Fig. 9. Transition system branching bisimilar to infinitely branching one

Now, checking all configurations we can see this transition system is generated by the pushdown automaton in Figure 10. This pushdown automaton is not popchoice-free.

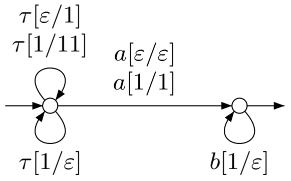


Fig. 10. Pushdown automaton that is not popchoice-free

5 Computable Processes

We proceed to give a definition of a Turing machine that we can use to generate a transition system. The classical definition of a Turing machine uses the memory tape to hold the input string at start up. We cannot use this simplifying trick, as we do not want to fix the input string beforehand, but want to be able to input symbols one symbol at a time. Therefore, we make an adaptation of a so-called *off-line* Turing machine, which starts out with an empty memory tape, and can take an input symbol one at a time.

Definition 8 (Reactive Turing Machine). *A* Reactive Turing Machine M *is defined as a six-tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ *where:*

1. *$\mathcal{S}$ is a finite set of states,*
2. *$\mathcal{A}$ is a finite action alphabet, $\mathcal{A}_\tau$ also includes the silent step τ,*

3. *$\mathcal{D}$ is a finite data alphabet, we add a special symbol $\Box$ standing for a blank and put $\mathcal{D}_\Box = \mathcal{D} \cup \{\Box\}$,*
4. *$\rightarrow \subseteq \mathcal{S} \times \mathcal{A}_\tau \times \mathcal{D}_\Box \times \mathcal{D}_\Box \times \{L, R\} \times \mathcal{S}$ is a finite set of* transitions *or* steps*,*
5. *$\uparrow \in \mathcal{S}$ is the initial state,*
6. *$\downarrow \subseteq \mathcal{S}$ is the set of final states.*

If $(s, a, d, e, M, t) \in \rightarrow$, we write $s \xrightarrow{a[d/e]M} t$, and this means that the machine, when it is in state s and reading symbol d on the tape, will execute input action a, change the symbol on the tape to e, will move one step left if $M = L$ and right if $M = R$ and thereby move to state t. It is also possible that d and/or e is $\Box$: if d is $\Box$, the reading head is looking at an empty cell on the tape and writes e; if e is $\Box$ and d is not, then d is erased, leaving an empty cell. At the start of a Turing machine computation, we will assume the Turing machine is in the initial state, and that the memory tape is empty (only contains blanks).

By looking at all possible executions, we can define the transition system of a Turing machine. The states of this transition system are the configurations of the Reactive Turing Machine, consisting of a state, the current tape contents, and the position of the read/write head. We represent the tape contents by an element of $\mathcal{D}_\Box^*$, replacing exactly one occurrence of a tape symbol d by a marked symbol $\bar{d}$, indicating that the read/write head is on this symbol. We denote by $\bar{\mathcal{D}}_\Box = \{\bar{d} \mid d \in \mathcal{D}_\Box\}$ the set of marked tape symbols; a *tape instance* is a sequence $\delta \in (\mathcal{D}_\Box \cup \bar{\mathcal{D}}_\Box)$ such that δ contains exactly one element of $\bar{\mathcal{D}}_\Box$.

A tape instance thus is a finite sequence of symbols that represents the contents of a two-way infinite tape. We do not distinguish between tape instances that are equal modulo the addition or removal of extra occurrences of a blank at the left or right extremes of the sequence. The set of configurations of a Reactive Turing Machine now consists of pairs of a state and a tape instance. In order to concisely describe the semantics of a Reactive Turing Machine in terms of transition systems on configurations, we use some additional notation.

If $\delta \in \mathcal{D}_\Box$, then δ^{-} is the tape instance obtained by placing the marker on the right-most symbol of δ if this exists, and $\bar{\Box}$ otherwise. Likewise, ${}^{-}\delta$ is the tape instance obtained by placing the marker on the left-most symbol of δ if this exists, and $\bar{\Box}$ otherwise.

Definition 9. *Let $M = (\mathcal{S}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ be a Turing machine. The* labeled transition system *of M, $\mathcal{T}(M)$, is defined as follows:*

1. *The set of states is the set of configurations $\{(s, \delta) \mid s \in \mathcal{S}, \delta$ a tape instance$\}$.*
2. *The transition relation $\rightarrow$ is the least relation satisfying, for all $a \in \mathcal{A}_\tau, d, e \in \mathcal{D}_\Box, \delta, \zeta \in \mathcal{D}_\Box^*$:*
   - $(s, \delta\bar{d}\zeta) \xrightarrow{a} (t, \delta^{-}e\zeta)$ *iff* $s \xrightarrow{a[d/e]L} t$,
   - $(s, \delta\bar{d}\zeta) \xrightarrow{a} (t, \delta e{}^{-}\zeta)$ *iff* $s \xrightarrow{a[d/e]R} t$.
3. *The initial state is $(\uparrow, \bar{\Box})$;*
4. *$(s, \delta) \downarrow$ iff $s \downarrow$.*

Now we define an *executable process* as the branching bisimulation equivalence class of a transition system of a Reactive Turing Machine.

As an example of a Reactive Turing Machine, we define the (first-in first-out) queue over a data set $\mathcal{D}$. It has the initial and final state at the head of the queue. There, output of the value at the head can be given, after which one move to the left occurs. If an input comes, then the position travels to the left until a free position is reached, where the value input is stored, after which the position travels to the right until the head is reached again. We show the Turing machine in Figure 11 in case $\mathcal{D} = \{0, 1\}$. A label containing an n, like $\tau[n/n]L$ means there are two labels $\tau[0/0]L$ and $\tau[1/1]L$.

The queue process is an executable process, but not a pushdown process.

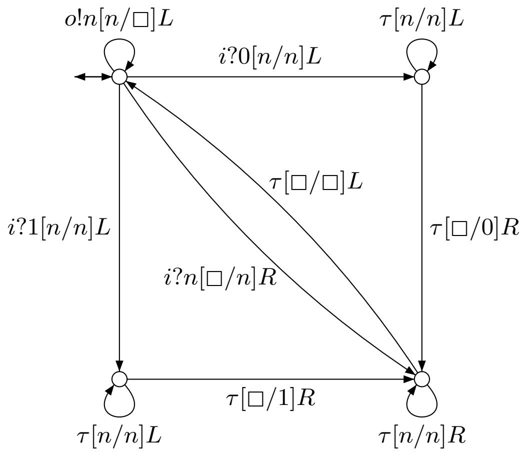


Fig. 11. Reactive Turing Machine for the FIFO queue

We call a transition system *computable* if it is finitely branching and there is a coding of the states such that the set of final states is decidable and for each state, we can determine the set of outgoing transitions. A transition system is *effective* if its set of transitions and set of final states are recursively enumerable. The following results are in [6].

Theorem 3. *The transition system defined by a Reactive Turing Machine is computable.*

Theorem 4. *Every effective transition system is branching bisimilar with a transition system of a Reactive Turing Machine.*

As in the case of the pushdown automaton, we can make the interaction between the finite control and the memory explicit, and turn this into a recursive specification.

Theorem 5. *For every Reactive Turing Machine M there exists a regular process expression p and a linear recursive specification E, and for every regular* *process expression p and linear recursive specification there exists a Reactive Turing Machine M such that*

$$\mathcal{T}(M) \leftrightarrows_b \mathcal{T}_{E \cup E_Q}(\tau_{i,o}(\partial_{i,o}(p \parallel Q^{io}))).$$

In this theorem, we use the queue process as defined above, and its specification E_Q to be defined next. By putting a finite control on top of a queue, we can simulate the tape process of a Reactive Turing Machine. The control of the Turing machine together with this control, can be specified as a finite-state process.

We finish by giving a finite recursive specification E_Q for a queue with input channel i and output channel o, Q^{io}, using all syntax elements of our process theory TCP$^*_\tau$ except for sequential composition and iteration. It uses an auxiliary channel l, and six interrelated equations.

$$Q^{io} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} i?d.\tau_l(\partial_l(Q^{il} \parallel (\mathbf{1} + o!d.Q^{lo})))$$

$$Q^{il} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} i?d.\tau_o(\partial_o(Q^{io} \parallel (\mathbf{1} + l!d.Q^{ol})))$$

$$Q^{lo} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} l?d.\tau_i(\partial_i(Q^{li} \parallel (\mathbf{1} + o!d.Q^{io})))$$

$$Q^{ol} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} o?d.\tau_i(\partial_i(Q^{oi} \parallel (\mathbf{1} + l!d.Q^{il})))$$

$$Q^{li} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} l?d.\tau_o(\partial_o(Q^{lo} \parallel (\mathbf{1} + i!d.Q^{oi})))$$

$$Q^{oi} \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in \mathcal{D}} o?d.\tau_l(\partial_l(Q^{ol} \parallel (\mathbf{1} + i!d.Q^{li})))$$

Now, the theorem above implies that recursive specifications over our syntax (even omitting sequential composition and iteration) constitute a grammar for all executable processes.

6 Conclusion

We established the notion of an execution in this paper, that enhances a computation by taking interaction into account. We do this by marrying computability theory, moving up from finite automata through pushdown automata to Turing machines, with concurrency theory, not using language equivalence but branching bisimilarity on automata.

Every undergraduate curriculum in computer science contains a course on automata theory and formal languages. On the other hand, an introduction to concurrency theory is usually not given in the undergraduate program. Both theories as basic models of computation are part of the foundations of computer science. Automata theory and formal languages provide a model of computation where interaction is not taken into account, so a computer is considered as a stand-alone device executing batch processes. On the other hand, concurrency theory provides a model of computation where interaction is taken into account. Concurrency theory is sometimes called the theory of reactive processes.

Both theories can be integrated into one course in the undergraduate curriculum, providing students with the foundation of computing, see [1]. This paper provides a glimpse of what happens to the Chomsky hierarchy in a concurrency setting, taking a labeled transition system as a central notion, and dividing out bisimulation semantics on such transition systems.

References

1. Baeten, J.C.M.: Models of Computation: Automata and Processes. Technische Universiteit Eindhoven, Syllabus 2IT15 (2010)
2. Baeten, J.C.M., Basten, T., Reniers, M.A.: Process Algebra (Equational Theories of Communicating Processes). Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2009)
3. Baeten, J.C.M., Corradini, F., Grabmayer, C.A.: A characterization of regular expressions under bisimulation. Journal of the ACM 54(2):6, 1–28 (2007)
4. Baeten, J.C.M., Cuijpers, P.J.L., Luttik, B., van Tilburg, P.J.A.: A process-theoretic look at automata. In: Arbab, F., Sirjani, M. (eds.) FSEN 2009. LNCS, vol. 5961, pp. 1–33. Springer, Heidelberg (2010)
5. Baeten, J.C.M., Cuijpers, P.J.L., van Tilburg, P.J.A.: A context-free process as a pushdown automaton. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 98–113. Springer, Heidelberg (2008)
6. Baeten, J.C.M., Luttik, B., van Tilburg, P.J.A.: Reactive Turing machines. Draft (2010)
7. Baeten, J., Luttik, B., Muller, T., van Tilburg, P.: Expressiveness modulo bisimilarity of regular expressions with parallel composition. In: Fröschle, S., Valencia, F.D. (eds.) Proceedings EXPRESS 2010, number xx in EPTCS, pp. 229–243 (2010)
8. Basten, T.: Branching bisimilarity is an equivalence indeed! Information Processing Letters 58(3), 141–147 (1996)
9. van Glabbeek, R.J.: What is Branching Time Semantics and why to use it? Bulletin of the EATCS 53, 190–198 (1994)
10. van Glabbeek, R.J.: The Linear Time – Branching Time Spectrum I. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, pp. 3–99. Elsevier, Amsterdam (2001)
11. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM 43(3), 555–600 (1996)
12. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Pearson, London (2006)
13. Milner, R.: A Calculus of Communication Systems. LNCS, vol. 92. Springer, Heidelberg (1980)
14. Moller, F.: Infinite results. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 195–216. Springer, Heidelberg (1996)
15. Park, D.M.R.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
16. Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebr. Program., 60-61, 17–139 (2004)
17. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society 42(2), 230–265 (1936)

Scribbling Interactions with a Formal Foundation*

Kohei Honda[1], Aybek Mukhamedov[1], Gary Brown[2], Tzu-Chun Chen[1], and Nobuko Yoshida[3]

[1]Queen Mary, University of London
[2]Red Hat, Inc.
[3]Imperial College London

Abstract. In this paper we discuss our ongoing endeavour to apply notations and algorithms based on the π-calculus and its theories for the development of large-scale distributed systems. The execution of a large-scale distributed system consists of many structured conversations (or sessions) whose protocols can be clearly and accurately specified using a theory of types for the π-calculus, called *session types*. The proposed methodology promotes a formally founded, and highly structured, development framework for modelling and building distributed applications, from high-level models to design and implementation to static checking to runtime validation. At the centre of this methodology is a formal description language for representing protocols for interactions, called Scribble. We illustrate the usage and theoretical basis of this language through use cases from different application domains.

1 Introduction

A fundamental challenge in modern computing is the establishment of an effective and widely applicable development methodologies for distributed applications, comparable in its usability to the traditional methodologies for non-distributed software built on, among others, core UML diagrams and object-oriented programming languages. Though a middle-to-large-scale application is almost always distributed nowadays, and in spite of the presence of an accelerating infrastructural support for portable and reliable distributed components through e.g. clouds, software developers (including architects, designers and programmers alike) are still lacking well-established development methodologies for building systems centring on distributed processes and their interactions. For example, there is no central computational abstractions (comparable to classes and objects) for capturing distributed interactions usable throughout development stages; no UML diagrams are widely in use for modelling distributed applications; no major programming languages offer high-level, type-safe communication primitives, leaving treatment of communications to low-level APIs. In short,

* This work is partially supported by EPSRC EP/F003757, EP/F002114, EP/G015635 and EP/G015481.

we are yet to have a general and tangible framework for developing distributed, communication-centred software systems.

We believe that one of the major reasons why it is so hard to even conceive an effective software development framework for distributed systems, is the lack of a core *descriptive framework*, with a uniform conceptual and formal foundation and usable throughout development stages. To illustrate this point, let us briefly examine the descriptive framework in one of the traditional development methodologies for non-distributed software, underpinned by UML diagrams and object-oriented programming. In this framework, the description of computation centres on *objects* (which belong to *classes*) and *operations on objects*, a representative paradigm of sequential computation. Class Diagram in UML and all associated core modelling diagrams such as Sequence Diagrams and State Charts follow this paradigm; and it is supported by many high-level programming languages, including Java, C++, C$\sharp$ and Python.[1]

Types and logics, two linchpins of theories of computing, play fundamental and mutually enriching roles in this traditional descriptive framework. Types, as seen in the now familiar APIs, offer a basic notion of *interface* of subsystems, tightly coupled with the central computational dynamics of this paradigm, i.e. invoking objects and returning results. This dynamics is embodied in high-level programming primitives, which in turn enables cheap and compositional static validation at the compile time [15, 29], leading to modular software development. Types are also a basis of logical specifications. In the widely practiced modelling framework known as Design-by-Contracts (DbC) [28], assertions elaborate types with predicates. Assertions are expressive, allowing us to pinpoint practically any property one wishes to specify [21], though automatic validation is not always possible. Assertions offer a refined form of modular software development through compositional behavioural contracts.

In the light of this well-established (and highly successful) engineering framework in the traditional development methodology, a natural question is whether we can build its analogue in the world of distributed processes, centred on common high-level abstraction for modelling and programming, and aiding modular software development on a rigorous theoretical basis.

This paper illustrates our ongoing endeavour to build a core descriptive framework and the associated development environment for large-scale distributed systems based on the π-calculus [30], centring on a simple language for describing interactions, called Scribble. A key insight is that a distributed system can be naturally and effectively articulated as a collection of possibly overlapping structured conversations, and that the structures of these conversations, or *protocols*, can be clearly and accurately described using a type theory of the π-calculus, called *session types* [23, 24, 39]. In Section 2, we discuss how protocols play a fundamental role for modelling and building distributed applications in diverse domains. In Section 3, we introduce Scribble. In Section 4, we present larger description examples in Scribble from real-world use cases. Section 5 outlines

[1] Having a different origin, functional languages such as Haskell and ML share a common paradigm, data belonging to data types and operations on data.

a theoretical basis of Scribble. Section 6 discusses a development framework. Section 7 concludes with related and future work.

2 Background: Modelling Interactions through Protocols

Protocols in Interactional Computing. The idea of protocols becomes important for general software development when the shape of software becomes predominantly a collection of numerous distributed processes communicating with each other. Such *interactional computing* is increasingly common in practice, from web services to financial protocols to services in clouds to parallel algorithms to multicore chips. Processes will be engaged in many interleaving conversations, each obeying a distinct protocol: the aggregate of overlapping conversations make up a whole distributed system. Dividing the design into distinct conversations promote tractability because the structure of one conversation in an application are relatively unaffected by other conversations.

A protocol offers an agreement on the ways interactions proceed among two or more participants. Without such an agreement, it is hard to do meaningful interactions: participants cannot communicate effectively, since they do not know when the other parties will send what kind of data and through which channels. This is why the need to describe protocols have been observed in many different contexts in the practice of interactional computing, as we illustrate below.

Needs for Protocols (1): Global Financial Network. ISO TC68, the Technical Committee for Global Financial Services in ISO, recognized the need for a mechanism to register and maintain international financial protocols (FPs) under the auspice of ISO. This has led to the establishment of a working group for FPs, WG4, which is in charge of drafting the evolving global standard for FPs, ISO20022 [41], using high-level models for describing message formats based on UML. The use of high-level models enable flexible engineering, such as compilation to different document format (e.g. XML schemas and ASN.1), semantic matching of message fields, and model-driven development. [20]

However a message format alone cannot describe an FP in its entirety: the flows in which asynchronous messages are exchanged is at the heart of the FPs. In ISO20022, this dynamic aspect of a FP is called its *message choreography* ("Every Business Transaction contains its own Message Choreography" [41]). In spite of its importance, the chair of WG4 observed that the description of the message choreography through the current technology has severe limitations:

1. *It is imprecise:* The descriptions of protocols are unclear, ambiguous and misleading, and legally unusable.
2. *It is incomplete:* It is impossible to describe the structure and constraints of FPs in their entirety up to a suitable abstraction level.
3. *It is informal:* The description cannot be used for formal reasoning about protocols; for checking their internal consistency; for verifying, either by hand or by machine, the conformance of endpoint programs against a given protocol; for code generation; for testing; and for runtime control.

A precise, complete and formal description of message choreography would offer a vital tool for harnessing and governing global FPs. A long-term goal of WG4 is to identify an effective method for describing message choreography of FPs, and use it in future versions of ISO20022.

Needs for Protocols (2): Operating System for Multicore CPUs. We turn our eyes to a basic form of systems software, operating systems. Most commodity computers nowadays are equipped with multi-core processors, which offer an effective way of harnessing high-density transistor circuits without incurring the performance penalties associated with monolithic processors. This trend is expected to continue in future, where many-core processors, whose numerous cores share a high-bandwidth on-chip interconnect, will become a commonplace [5]. As a consequence, computers are increasingly resembling a distributed system, which cannot be effectively utilised by traditional monolithic OS kernels built around shared data structures that suffer from performance and scalability issues in a parallel execution environment.

Barrelfish is a new multi-kernel OS architecture that aims to address the challenge [4]. It is designed to run on heterogeneous multicore machines and is structured as a distributed system of cores that communicate via explicit message passing and share no memory. Early benchmarks on present day multicore computers showed that the performance of Barrelfish is comparable to that of existing commodity operating systems and can scale better to support future many-core hardware [4]. There are clear parallels between Barrelfish OS and a distributed system, and in particular, the importance of having unambiguous specification of communication protocols. However, the current programming development for Barrelfish only offers description of procedural interface, making it hard to ensure compatibility at the level of asynchronous message passing among OS components, a predominant mode in this operating system.

Needs for Protocols (3): Web Services. In web services, applications make an extensive use of communications among components and services through the standardised format and transport technologies (e.g. URI, XML and TCP/HTTP), increasingly combined with other distributed computing technologies such as clouds, messaging and distributed store. Business transactions using web services are often termed *business protocols* because each of them obeys an agreed-upon conversation structure. Web Services Choreography Description Language (WS-CDL) [13] was conceived in W3C as a declarative, XML-based domain-specific language for specifying business protocols. It is also a first standardization effort done in collaboration with the π-calculus experts from academia.

WS-CDL is notable in that its description captures "global" ordering – a *choreography* – of observable behaviour of participants in a channel-based communication. It comprises a rich set of concepts (roles, work units, exceptions, etc.) and general control constructs (sequencing, parallel, conditionals, recursion) for expressing multi-party interaction. At the same time, as a descriptive means for protocols, it has several drawbacks: first, although a subset of

WS-CDL has been given a formal semantics using the π-calculus [12], the language as a whole is not equipped with the notion of *projection from a global specification to endpoint specifications* (which is important for deriving communication specification for local participants); and it lacks a clear stratification between specifications and executable programs.

Needs for Protocols (4): Large-scale Cyberinfrastructure. The Ocean Observatories Initiative (OOI) is a large-scale project funded by US National Science Foundation for implementation of a distributed environmental science observatory with persistent and interactive capabilities that have a global physical observatory footprint [14, 35]. A key component of the OOI is a comprehensive cyberinfrastructure (CI), whose design is based on loosely coupled distributed services and agents, expected to reside throughout the OOI observatories, from seafloor instruments to on-shore research stations. The CI acts as an integrating element that links the sub-networks of OOI into a coherent system-of-systems and uses a large catalogue of communication protocols among distributed instruments and stakeholders. These protocols are required to be unambiguously specified for the implementation and runtime communication monitoring.

Towards a Descriptive Basis for Protocols. The pervasiveness and complexity of interactional computation in modern and future computing highlight the need for a general and rigorous protocol description framework, usable throughout the software development life cycle, equipped with a clear, transparent semantic basis, and offering foundations for modular software development through computer-aided validation and verification tools. We now illustrate our recent efforts to develop such a framework, centred on a small description language for scribbling protocols.

3 Overview of Scribble

The goal of Scribble is to provide a formal and yet intuitive language and tools for specifying and reasoning about communication protocols and their implementations, based upon the theory of multiparty session types [6, 24, 43]. Figure 1 gives an overview of this software framework, which we call the *Scribble framework*.

Applications can implement interaction behaviour through the conversation API, a high-level language-independent message-passing interface, to be realized in various high-level programming languages (such as ML, Java, Python, C♯, C++ and others). Static validation is carried out with the aid of a conversation API. In place of the conversation API, we can also use language extensions with intrinsic type checking capability, as studied in [25, 26]. The protocol typechecker inspects the application code and decides whether its communication behaviour in a conversation follows the prescribed protocol. Dynamic validation is performed by a monitor that reads in a Scribble protocol specification and inspects runtime communication behaviour of an application. The monitor checks that its interaction follows the behaviour of the corresponding role(s) prescribed by the protocol. For further discussions, see Section 5.

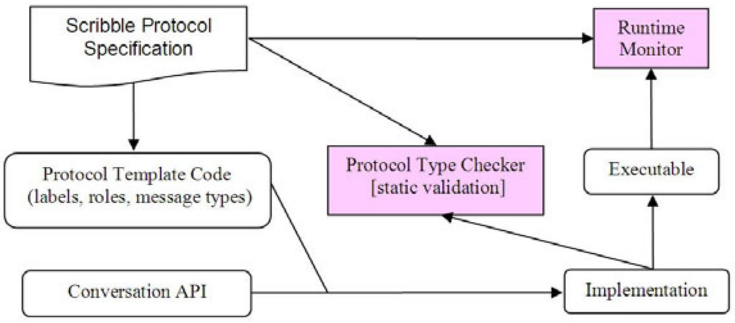

Scribble Protocol Specification
Runtime Monitor
Protocol Template Code (labels, roles, message types)
Protocol Type Checker [static validation]
Executable
Conversation API
Implementation

Fig. 1. Scribble framework overview

The Scribble framework is currently a work in progress. In the following we present an overview of its underlying protocol specification language, Scribble.

Hello World. We start the overview of Scribble with a customary hello-world example as a protocol, illustrating its basic structure.

```
import Message;

protocol GreetWorld {
  role You, World;
  greet(Message) from You to World;
}
```

The above protocol definition intuitively says:

> The protocol uses `Message` type defined using the `import` statement. Each conversation instance (a run) of the protocol involves two participants – one taking the role `You` and the other taking the role `World`. In each conversation instance, `You` sends a single message to `World`, which consists of the operation name `greet` with a value of type `Message`.

This protocol uses a single interaction (more complex examples will appear later). A *protocol* such as `GreetWorld` gives a global description of interactions among two or more participants. A *session*, or *conversation*, is an instantiation of a protocol that follows the protocol's rules of engagement. *Principals* represent entities, such as corporations and individuals, who are responsible for performing communication actions in distributed applications. When a principal participates in a conversation (i.e. becoming its *participant*), it does so by taking up specific role(s) stipulated in the underlying protocol.

Transport Characteristics. We assume the following properties of the underlying message transport. This is an important assumption to understand the semantics of Scribble.

- *Asynchrony*: send actions are non-blocking.
- *Message order preservation*: the order of messages from the same participant to another participant in a single conversation is preserved.
- *Reliability*: a message is never lost or tampered with during transmission.

These properties may be realised by a transport layer possibly combined with runtime systems at endpoints. They are natural assumptions for many existing transports, be it in Internet, high-performance LAN or on-chip interconnect.

Main Constructs. The top-level grammar of a Scribble description comprises:

(1) At most one *preamble*, which consists of one or more `import` statements: in the `GreetWorld` protocol we have just seen, this is Line 1, importing a message type called `Message`.

(2) A single *protocol definition* (Lines 3–6 in the `GreetWorld` protocol), which consists of the keyword protocol, the name of the protocol (e.g. `GreetWorld`), and the main part – the *protocol body* – enclosed by curly braces.

The protocol body consists of one or more *role declarations* followed by *interaction description.* Roles are placeholders for participating endpoints. When a protocol is instantiated to a concrete conversation, each role, say `You`, is *bound* to a principal, making the latter a participant in that conversation. The behaviour of this participant should follow that of `You` prescribed in the protocol. In the `GreetWorld` protocol, Line 4 gives *role declarations*, which specifies that the protocol description includes two endpoints, `You` and `World`. The grammar of the role declaration is:

```
role role1, ..., roleN;
```

where `roleNamei` is a role name. This is equivalent to:

```
role role1;
...
role roleN;
```

All role names should be distinct and the order does not matter.

The *interaction description* is the main part of the protocol description. There is at most one such description in a protocol specification. It specifies one or more interactions, which belong to a syntactic category called *interaction sentence.*

The grammar of interaction sentences has several forms. Below we describe a few basic types that appear in the current version of Scribble:

(1) Interaction, of the form:

```
msgType from role1 to role2;
```

which defines an *interaction signature* (often simply *interaction*), and reads:

> A participant playing the role `role1` sends a message of type `msgType` to a participant playing the role `role2` and the latter eventually receives the message.

Above the message type `msgType` is imported in an enclosing environment, and can be a base or a composite type. Base type can be a primitive type common to many programming languages, such as `int`, `bool`, or a user-defined type. In the current syntax, composite message types are restricted to an operator name applied to (possibly empty) sequence of base message types: `OpName(ValType1, .., ValTypeN)`. Operator names give clarity to interaction signatures, just as object methods determine its interaction signature in object-oriented programming.

(2) Sequencing, of the form:

```
I1; I2; ...; In
```

represents an interaction sentence, where if the *same role name* appears in both `Ii` and `I(i + k)`, then the interaction actions of that participant take place under a temporal order (thus if none of the role names overlap between `I1` and `In`, then no order is specified). This interpretation is faithful to the asynchronous semantics of communications (formally treated in [24]). For example, in:

```
1   order(Goods) from Buyer to Seller;
2   deliver(Shipment) from Seller to Supplier;
3   confirm(Invoice) from Seller to Buyer;
```

`Seller` sends an invoice to `Buyer` (Line 3) only after it receives an order from `Buyer` and sends a shipment order to Supplier (Lines 1, 2). `Buyer` expects an invoice from `Seller` after it sends an order to `Seller`.

(3) Unordered (also called **Parallel**), of the form:

```
I1 & I2 & ... In
```

represents interleaved interactions that may be observed in any order. We write:

```
msgType from role1 to role2,.., roleK;
```

for a shorthand of:

```
msgType from role1 to role2 & .. & msgType from role1 to roleK;
```

(4) Directed Choice, of the form:

```
choice from role1 to role2,.., roleK {
  msgType1: I1
  ..
  msgTypen: In
}
```

represents interaction flow branching, where role1 makes a choice `msgTypej` to continue interaction following scenario in `Ij`. For example, in :

```
order(Goods) from Buyer to Seller;
choice from Seller to Buyer {
  accept(Invoice):
    payment(CardDetails) from Buyer to Seller;
  decline():
    end;
}
```

After Buyer sends an order to `Seller`, `Seller` makes a choice whether to accepts it or not. If it decides the former, `Seller` returns an invoice to `Buyer` and subsequently waits for a payment in return from him.

(5) Recursion, of the form:

```
rec BlockName { I }
```

where #BlockName appears inside I at least once, signifying a repetition of the whole block when #BlockName is encountered. For example, in

```
rec X {
  order(Goods) from Buyer to Seller;
  choice from Seller to Buyer {
    accept():
      ..
      #X;
    decline():
      end;
  }
}
```

Seller can continuously accept orders from Buyer, until it decides to decline one. When Seller declines an order, the repetition stops.

(6) Nested protocol, of the form:

```
run Protocol(param1,.., paramk, roleInChild1=roleInParent1,.., roleInChildn=
      roleInParentn);
```

represents protocol nesting. When run directive is encountered in the interaction flow of a conversation, a new conversation is instantiated and followed as prescribed by the nested Protocol. The Protocol may require positional arguments, as well as role keyword arguments, by which the roles in the Protocol are instantiated with the roles of the enclosing protocol.

Scribble includes other forms of interaction sentences (global escape, delegation, repetition, etc), which we omit for brevity of this presentation, see [37].

4 Scribble Examples

Scribble can be utilised to express communication protocols from a wide range of application domains. In this section we present two examples, taken from web services [13] and from multikernel OS [4, 40].

Web services: Travel Agent. Travel Agent is an interaction scenario designed by the WS-CDL Working Group [13], intended to represent general concepts common to many applications of web services. It comprises multiple participants – a client, a travel agent and a number of service providers – and involves complex branching and repetition in the interaction flow. Figure 2 gives an informal description of the interaction behaviour among the participants.

We present a specification of Travel Agent protocol in Scribble in two parts with: ReserveTravel protocol (Figure 3), by which the client enquires about and reserves travel services with the help of an agent, and PurchaseTravel protocol (Figure 4) for subsequent service booking interaction. PurchaseTravel is parametric in the number of service providers that the agent communicates with in the preceding ReserveTravel protocol. The specification makes use of recursion to

1. The client interacts with the travel agent to request information about various services.
2. Prices and availability matching the client requests are returned to the client. The client can then perform one of the following actions:
   (a) The client can refine their request for information, possibly selecting more services from the provider (Repeat step 2). OR
   (b) The client may reserve services based on the response, OR
   (c) The client may quit the interaction with the travel agent.
3. When a customer makes a reservation, the travel agent then checks the availability of the requested services with each service provider.
4. Either
   (a) All services are available, in which case they are reserved. OR
   (b) For those services that are not available, the client is informed.
      - Either
         i. Given alternative options for those services. OR
         ii. Client is advised to restart the search by going back to step 1.
      - Go back to step 3.
5. For every relevant reserved service the travel agent takes a payment for the reservation (credit card can be used as a form of payment)
6. The client is then issued a reservation number to confirm the transaction.
7. Between the reservation and the final date of confirmation, the client may modify the reservation. Modifications may include cancellation of some services or the addition of extra services.

Fig. 2. Travel Agent protocol: informal description

```
import TravelAgent.messages.*;

protocol ReserveTravel {
  role Client, Agent, Provider[1..num_providers];

  query(Services) from Client to Agent;
  Services_info from Agent to Client;

  rec X {
    choice from Client to Agent {
      more_info():
        query(Services) from Client to Agent;
        Services_info from Agent to Client;
        #X;
      reserve():
        query(Services) from Agent to Provider[1..num_providers];
        Services_info from Provider[1..num_providers] to Agent;
        choice from Agent to Client {
          all_available():
            reserve(Services) from Agent to Provider[1..num_providers];
            run PurchaseTravel(num_providers);
          altern_services():
            Altern_services_info from Agent to Client;
            #X;
          restart(): end;
        }
      quit(): end;
    }
  }
}
```

Fig. 3. Travel Agent: ReserveTravel protocol in Scribble

```
import TravelAgent.messages.*;

protocol PurchaseTravel(num_providers) {
  rec X {
    choice from Client to Agent {
      cancel():
        cancel(Services) from Client to Agent;
        #X;
      add_services():
        request(Extra_services) from Client to Agent;
        Extra_services_response from Agent to Client;
        #X;
      book():
        Payment from Client to Agent;
        book(Services) from Agent to Provider[1..num_providers];
        confirm(Services) from Provider[1..num_providers] to Agent;
        choice from Agent to Client {
          confirm():
            Receipt from Agent to Client;
          timeout_error():
            Error_details from Agent to Client;
        }
      quit(): end;
    }
  }
}
```

Fig. 4. Travel Agent: PurchaseTravel protocol in Scribble

represent arbitrary repetition of a series of interactions. In ReserveTravel protocol, lines 6-13 correspond to steps 1 and 2a in Figure 2, lines 15-21 to steps 2b, 3 and 4a. Line 21 uses a nested protocol, `PurchaseTravel`, which describes a successful purchase of travel services by the client (steps 5, 6 and 7 of Figure 2).

Multikernel OS: Distributed USB Manager. Next we present a protocol from a distributed USB manager in Barrelfish multi-kernel OS [4, 40], consisting of three primary modules that cooperate via explicit message passing:

– *EHCI host controller driver (HCD).* The host driver manages interaction with the host controller hardware and provides a high-level interface for communicating with the hardware.
– *Client device driver.* The client driver carries out interaction with a USB device and exposes services of the device to applications.
– *USB manager.* The manager is responsible for coordination of the modules and allocation of resources.

The USB manager has the most complex communication logic among these modules, performing orchestration of other components. In Figure 5 we informally describe a USB device `Plug_Unplug` protocol.

Figure 6 presents a specification of `Plug_Unplug` protocol in Scribble. The interaction has a linear structure and its subtlety lies in the correct interleaving of control messages (`ctrl_exe`) with data commands (`dctrl_exe`) between the USB manager and HCD.

- Either, a new device is inserted into one of the USB ports:
  1. HCD notifies the USB manager that a device is inserted.
  2. The manager reads the USB device descriptor (via HCD) that contains the number of configurations, device protocol, class and other information.
  3. The manager reads each configuration reported by the above descriptor, which contain information about power requirements, interfaces and endpoints. The configurations are read twice: at first, to determine the total length of data needed to read interface and endpoint descriptors and subsequently to fetch all interface and endpoint descriptors.
  4. The manager switches the device into addressed mode and crosschecks that by reading device descriptor again.
  5. The manager assigns a configuration and interfaces to the device, which can be later changed by device driver.
  6. The manager locates appropriate client driver by querying System Knowledge Base (SKB). If a match is found, SKB returns the server name running the required driver.
  7. The manager probes the driver if it accepts the new device or not.
     - If the driver accepts the request, it requests the manager to establish a logical connection (pipe) with the device. The pipe is subsequently controlled by HCD.
     - If the driver rejects the request, the manager cleans up its resources.
- OR, a USB device is removed from a port:
  1. HCD notifies USB manager the device is removed.
  2. The manager cleans up its resources and notifies the client driver.

Fig. 5. Distributed USB PlugUnplug protocol from Barrelfush multi-kernel OS

```
import PlugUnplug.messages.*;

protocol PlugUnplug {
  role HCD, USB_Manager, Driver, SKB;

  choice from HCD to USB_Manager {
    notify_new_device(port):
      // step 2
      dctrl_exe(req,buf,sz,addr,id) from USB_Manager to HCD;
      dctrl_done(id) from HCD to USB_Manager;
      // step 3
      dctrl_exe(req,buf,sz,addr,id) from USB_Manager to HCD;
      dctrl_done(id) from HCD to USB_Manager;
      dctrl_exe(req,buf,sz,addr,id) from USB_Manager to HCD;
      dctrl_done(id) from HCD to USB_Manager;
      // step 4
      ctrl_exe(req,dev,id) from USB_Manager to HCD;
      ctrl_done(id) from HCD to USB_Manager;
      dctrl_exe(req,buf,sz,addr,id) from USB_Manager to HCD;
      dctrl_done(id) from HCD to USB_Manager;
      // step 5
      ctrl_exe(req,dev,id) from USB_Manager to HCD;
      ctrl_done(id) from HCD to USB_Manager;
      // step 6
      get_addr(dev,buf,id) from USB_Manager to SKB;
      get_addr_done(id) from SKB to USB_Manager;
      // step 7
      probe(dev,class,prot) from USB_Manager to Driver;
      choice from Driver to USB_Manager {
        probe_done(ACCEPT,dev):
          pipe_req(dev,type,dir) from Driver to USB_Manager;
          pipe_resp(resp,pipe) from USB_Manager to Driver;
        probe_done(REJECT,dev):
          // clean-up
      }
    notify_device_removal(port):
      disconnect(dev) from USB_Manager to Driver;
  }
}
```

Fig. 6. PlugUnplug protocol in Scribble

5 Formal Foundations of Scribble

General Ideas. Scribble is formally based on the π-calculus and its type theory called *session types*. Having a general, well-understood theoretical foundation is important since without such a foundation, we cannot establish clear semantics for protocol descriptions, we cannot rigorously analyse how descriptions relate to dynamics, and we cannot accurately state and validate properties of a target system. The π-calculus enjoys full expressiveness for representing interactional behaviours in spite of its tiny syntax: it can mathematically embed a large class of communication-centred software behaviours, including those of existing programming languages, without losing precision. For this reason, the study of session types in the π-calculus, including various validation algorithms, can be directly applicable to real-world programming languages. Below we informally outline the correspondence between the theory of session types and Scribble, including assurance of properties founded on this theory.

Types for Protocols: Session Types. In sequential programming language such as Java and C, a type mainly stipulates the data type of a variable. In particular, in typed languages, all variables should first be declared before they can be used. For example,

$$\mathsf{int}\ \ \mathsf{storage} = 1;$$

This program involves stating the type and name of a variable, and telling program that a field named storage exists, holds numerical data, and has an initial value of 1.

Extending this view to interactional computing, session types set the rules for a session (conversation), ensuring safe interactional behaviours for each session. The specification starts from a *global session type* or a *global type* [24], which describes the whole conversation scenario. It gives a specification for the whole protocol from a bird's eyes by giving the rules of conversations for all participants. This global type corresponds to a protocol in Scribble: the Scribble's protocol notation was born from the theories of global types studied in [6], which is the advancement of the theory presented in [24].

A local type represents the type of interactions for each role, played by a principal in a conversation. It is given by projecting the corresponding global type onto a specific role. The following example shows that, in a Buyer-Seller-Broker session s, a Buyer asks Broker for a product, and sends the product's name. Broker refers this request to Seller, then Seller replies to the Broker with the product's price. Broker refers the price to Buyer after receiving it.

$$\begin{aligned} G = \ & \mathsf{Buyer} \to \mathsf{Broker} : \mathsf{string}. \\ & \mathsf{Broker} \to \mathsf{Seller} : \mathsf{string}. \\ & \mathsf{Seller} \to \mathsf{Broker} : \mathsf{int}. \\ & \mathsf{Broker} \to \mathsf{Buyer} : \mathsf{int}. \\ & \mathsf{end}. \end{aligned}$$

The local types for Buyer are

$$\langle \mathsf{Broker} \rangle ! \langle \mathsf{string} \rangle . \langle \mathsf{Broker} \rangle ?(\mathsf{int}),$$

which are the projection from G onto Buyer. This local types indicate that a Buyer should firstly send a name of type string to Broker, and then wait to receive a price, which is a variable of type int, sent from Broker.

Similarly, the local type for Seller is given as

$$\langle \mathsf{Broker} \rangle ?(\mathsf{string}).\langle \mathsf{Broker} \rangle !\langle \mathsf{int} \rangle,$$

and the local type for Broker is

$$\langle \mathsf{Buyer} \rangle ?(\mathsf{string}).\langle \mathsf{Seller} \rangle !\langle \mathsf{string} \rangle.\langle \mathsf{Seller} \rangle ?(\mathsf{int}).\langle \mathsf{Buyer} \rangle !\langle \mathsf{int} \rangle.$$

Through a global type, we can stipulate the whole set of rules of interactional behaviours participated by all participants; whereas a local type enables the corresponding endpoint (local program) to know the rules of behaviours for a specific role in a conversation.

Safety Assurance by Session Types. The typing system of Scribble for multi-party sessions follows [6], using their global types and projection rules. A well-designed typing system can ensure error-free conversations among multi-party sessions [24], by typing each endpoint with the corresponding local type, which is projected from a stipulated global type. Thus, for a given conversation, we can assure each of its participants plays its role correctly. This assurance can be done effectively at the programming/compilation-time: we can derive a typing algorithm from the tying rules, which can (in)validate that a process, corresponding to an application program at some endpoint, is conforming to a projected local type. Thus session types provide static type-checking at the programming time.

When all endpoints are type-checked and they start interaction, they satisfy several significant properties. First, we have a formal theorem which says that

> "well-typed processes never exchange wrong values."

That is, if a process is expecting an integer, it will get an integer; and if it expects a string, it will receive a string. Secondly, inside each session (conversation), there is what is often called *linearity*:

> "an output is never shared by more than one inputs, and vice versa."

Finally, we can assure that the interactions through a well-typed conversation follow the initially stipulated protocol:

> "interactions inside a session among well-typed processes under a global type, never violate the scenarios given in that type."

This property can be further strengthened under certain conditions that interactions can always proceed in a session, so that they inevitably complete one of the scenarios given in a protocol, assuring an important liveness property. Here by a *liveness* property we mean a property demanding a process can surely do a good thing. In contrast, the preceding three properties are about *safety* since each says that a process never does a stipulated bad thing.

The type-based static checking (including projection) and properties ensured by the typing algorithm give a basis of diverse engineering practice and theories centring on session types. As has been studied in [8], a logical method, which elaborates session types with assertions (just as Design-by-Contract elaborates procedural types with logical formulae), can be built on this basis, which uses precisely the same framework except that it is lifted to logical elaboration of session types. Further, a series of studies show how we can consistently and effectively incorporate session types in the semantics and pragmatics of existing programming languages [19, 25, 26].

6 Development Framework

6.1 General Concepts

Project for Development Environment of Distributed Applications. For Scribble to be useful for development, it should be complemented with associated software tools including programming languages, integrated into a development environment. The present authors, in conversation and collaboration with academic and industry colleagues, started the design and implementation of core development tools centring on Scribble in late 2009, complemented by other activities. Our aim is to reach a simple and effective tool chain for the development of distributed applications which can effectively interface with existing artifacts and tools such as UML and Java. We are still in an early stage of design assessment and prototyping: below we illustrate some of its key ideas.

Modelling with Protocols. The requirement capture phase of the software development life cycle leads to the identification of significant scenarios, or use cases, associated with the target system's usage. For interactional systems, many of such use cases may as well be *conversational* — in the sense that they represent interactions among more than one actor. A use case can then be elaborated into one or more scenarios-as-conversations, each of which will obey a certain protocol: just as an object referred to in a use case scenario belongs to a class.

There are two functions which a tool can provide for this modelling stage: to **edit** protocols (with grammatical checks) and to **validate** their semantic consistency or conformance to other documents. To share protocols with other developers one can also **publish** protocols. One can also **project** a protocol with multiple participants to each endpoint (role), to produce a *local protocol* using the algorithm coming from the underlying theory (see Section 5). This gives a model of conversations from the local viewpoint.

Programming with Protocols. Protocols produced at the modelling stage may be refined into more concrete protocols at the design stage, so that they are eventually usable for implementation. A programmer may also need new protocols just for implementation purposes, as well as using already published protocols. She will then **edit a program**, which uses typed sessions for communications among programs, for example through a Scribble-aware API for communications. She can then statically **check conformance** of her program to stipulated protocols. Protocol descriptions can also be used to **test** programs, where interactions are checked against protocols.

At runtime, each endpoint application is executed through a runtime which links the high-level communication operations for sessions to the underlying messaging infrastructure [25, 26]. Multiple such endpoints will converse with each other over multiple conversations, where each endpoint participates in a session taking some role, as specified in the underlying protocol. Communications can be monitored to prevent a conversation from violating a protocol, at each endpoint and/or globally. If some anomaly is detected, a monitor will notify this fact to an entity (a virtual agent) in charge of policy enforcement. The protocol documents also form basic part of the design document of the system.

6.2 Concrete Design

Project for* Scribble*-based Development Environment. Scribble and associated tools are being developed by an open source project hosted at [37], with multiple academic and industry participants. Its purpose is to provide a collaborative environment to support the development of the language Scribble and associated tools. One of the exciting aspects of the project is the collaboration between academia and industry. Within the project, we are aiming to harness the best of these two worlds: leverage the academic results to develop cutting edge capabilities, while providing the stable software development life cycle required to deliver a higher quality and better supported product for use in industry.

Our current design of a tool chain focuses on Eclipse. To enable extensibility in this environment, we leverage the OSGi standard. The tooling includes an Eclipse-based context sensitive editor for Scribble. Since Eclipse is based on the OSGi framework, various Scribble-related functions become automatically available in the editor as OSGi modules become available. Extensibility also facilitates incorporation of new research ideas into existing tool functionalities.

Protocols are parsed by a parser generated by the ANTLR parser generator, producing an internal representation (a Java object model) through the abstract syntax tree. It is this representation which is used and acted upon by various validation modules, discussed next.

Static Validation and Other Algorithms. One of the fruits of the industry-academia collaboration in the Scribble project is the use of the latest research results on validation and other algorithms that are provably correct to give the required results. Each validation module is responsible for processing a Scribble protocol object model to output whether it is valid or not (with respect to a specific criteria). Validations can be arbitrarily chained, and are usually set up so that they are triggered automatically as a protocol description is created or updated in the editor. A new validation function can be added simply by installing an OSGi module implementing the appropriate interface. The main validation functions include:

1. Syntactic consistency (parsability)
2. Semantic consistency of session types (including linearity guarantee, which avoids a race condition in a conversation [24])
3. Conformance checking of a local protocol against a global one [31].

The conformance checking noted above is also used for local protocols extracted from an implementation of an endpoint in an already existing program, written in e.g. BPEL. These validations, together with other functions such as the projection of a global protocol to local protocols (using the latest algorithm from [43]), can be lifted to logical specifications following [8]. Another form of validation is dynamic validation through monitors, using an efficient internal representation of protocols. The design and implementation of these and other validation modules are under way, with a stable release planned in late 2011.

Finally, programming support for Scribble for existing languages comes from two sources: APIs and language extensions. In both cases, a type checker, which validate type correctness of programs against local protocols, plays a key role. Three implementations covering both approaches are under way, based on the latest research on session-based programming and runtime [25, 26].

7 Related Work and Conclusion

In this section we discuss some of the related work, with an emphasis on theoretical studies related to Scribble, and conclude with further topics.

Process Algebra. Process algebras, such as ACP [3], CSP [22] and π-calculus [30], present a semantic framework where interactional behaviour of software systems can be captured on a rigorous mathematical basis through a small set of operators for constructing processes. The fruits from the studies on these and other models of concurrency form an essential engineering foundation of Scribble. For example, behavioural equivalences such as bisimulations, a linchpin of theories of process algebras, offer the mathematical basis of key engineering activities such as optimisations, security, correctness of compiler, correctness of runtime, and various semi-automatic verifications.

Session Types and Other Description Frameworks. Session types [23, 39] have been studied over the last decade as a typed foundation for structured communication-centred programming using various programming languages and process calculi. The original binary session types have been generalised to *multiparty session types* [24], in order to guarantee stronger conformance to stipulated session structures when a protocol involves more than two parties. Theories of multiparty session types [6, 24] give the foundations of Scribble, together with their semantic basis given by the π-calculus. Validation and other algorithms from their studies are used as the core elements of its tool chain.

Since [24], the theory of multiparty session types has been extended in different directions, including a theory which ensures the progress property for interleaved multiparty sessions [6] (which also gave the formal basis of the Scribble's syntax); generalised type structures which allow communication optimisation through permutation [31]; and a static analysis for communication buffer overflows [16]. The existing notations for describing protocols include message sequence charts [9, 27] and UML sequence diagrams [34] (the latter when method calls are replaced by asynchronous signals). These notations are different in that they are not based on the abstraction of protocols as type signatures. Protocol descriptions from a different viewpoint are studied in [18], where one stipulates possible communication events among endpoints using a logic of commitment. WS-CDL [13] (discussed in Section 2) is one of the first expressive languages which allow description of interactions from a global viewpoint. WS-CDL is also a basis of the preceding validation tool by one of the authors (G.B.), pi4soa [36], on whose experience the design of the Scribble-based development environment is being carried out. In comparison with these descriptive languages, the multiparty generalisation of session types offer, for the first time, a framework of protocol descriptions where they are formally captured as type signature, together with a notion of type conformance through formal projection to endpoints.

Recently the theory of multiparty session types has been applied in different contexts, including protocol optimisation for distributed objects [38]; integrity of session interactions [7, 10]; type-safe asynchronous event programming [25]; safe and efficient parallel programming [32, 43]; multicore programming [44]; and medical guidelines [33]. Many of these studies are inspired by and/or inspire our industrial collaborations.

***Communication-Centred Programming Languages and* Scribble.** Occam-Pi [42] is a highly efficient systems-level concurrent programming language centring on synchronous communication channels, based on CSP and the π-calculus. Hewitt's Actor Model [1] is an influential programming model centring on asynchronous unordered message passing. Erlang [2] is a communication-centred programming language with emphasis on reliability based on actors. Scribble differs from these languages in that it is a protocol description language rather than a programming language, with formal foundations coming from the π-calculus and session types, intended to be used across multiple programming languages through different stages of software development.

Further Topics. To realise the full potential of the proposed approach in general and Scribble in particular, the incorporation of several recent advances of the theory of session types into the description language would be relevant. First, Scribble can be extended to the type-safe multiparty session exceptions recently developed in [11], in order to handle system failure and fault-tolerance in a larger class of distributed protocols, preserving type safety. The incorporation of the parametrised dependent type theory from [43] enables us to directly express more complex communication topologies. More recently, we studied a dynamic multirole session type in [17] where an arbitrary number of participants can dynamically join and leave an active session under a given role. This solved an open problem in the theory of multiparty session types, providing a new framework to handle common distributed communication patterns such as publisher-subscriber or P2P and chat protocols within the theory. The notion of the role with dynamic join capabilities in [17] was motivated by Scribble and protocol descriptions in it, demonstrating an interaction between practice centring on Scribble and academia brings a new theory which can be used to enrich type structures of Scribble. Another significant extension of the theory of multiparty session types, again motivated by a dialogue with practice, is our recent work [8] on an assertion framework built on session types. The incorporation of the assertion-based framework will enrich the expressiveness of Scribble as a tool for description by enabling the specification of fine-grained constraints. The framework generalises the traditional Design-by-Contract, offering a refined modular development framework for distributed communicating processes based on multiparty behavioural contracts.

Acknowledgements. We thank Raymond Hu, Matthew Rawlings and Munindar Singh for their comments. The Scribble project has been enriched by an extensive dialogue with academic and industry colleagues, including, but not limited to, Matthew Arrott, Matthew Rawlings, Steve Ross-Talbot and Olivier Pernet. The first author acknowledges a fruitful discussion with another industry colleague, Antony Alappatt. We thank all of them, and record that this series of a dialogue between practice and theories was made possible by the initiative and leadership of late Robin Milner, who is also the originator of many of the key theoretical foundations on which Scribble is based. For this reason, we dedicate this paper to him, and hope that our work from now on in this domain will contribute to the enrichment of both practice and theories of computing through their deep and honest dialogue, as Robin envisioned in his address in Edinburgh in 1986 and encouraged us in the early days of this endeavour.

References

1. Agha, G.: Actors: a model of concurrent computation in distributed systems. MIT Press, Cambridge (1986)
2. Armstrong, J.: Programming Erlang: Software for a Concurrent World. Pragmatic Bookshelf (2007)
3. Baeton, J., Wejland, W.: Process Algebra. Cambridge University Press, Cambridge (1990)
4. Baumann, A., et al.: The multikernel: a new os architecture for scalable multicore systems. In: SOSP, pp. 29–44. ACM, New York (2009)
5. Baumann, A., Peter, S., Schüpbach, A., Singhania, A., Roscoe, T., Barham, P., Isaacs, R.: Your computer is already a distributed system. why isn't your os? In: Proceedings of the 12th Conference on Hot Topics in Operating Systems, HotOS 2009, pp. 12. USENIX Association, Berkeley (2009)
6. Bettini, L., et al.: Global progress in dynamically interleaved multiparty sessions. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 418–433. Springer, Heidelberg (2008)
7. Bhargavan, K., Corin, R., Deniélou, P.-M., Fournet, C., Leifer, J.: Cryptographic protocol synthesis and verification for multiparty sessions. In: CSF, pp. 124–140 (2009)

8. Bocchi, L., Honda, K., Tuosto, E., Yoshida, N.: A theory of design-by-contract for distributed multiparty interactions. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 162–176. Springer, Heidelberg (2010)
9. Broy, M., Krüger, I.H., Meisinger, M.: A formal model of services. ACM Trans. Softw. Eng. Methodol. 16(1), 5 (2007)
10. Capecchi, S., Castellani, I., Dezani-Ciancaglini, M., Rezk, T.: Session Types for Access and Information Flow Control. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 237–252. Springer, Heidelberg (2010)
11. Capecchi, S., Giachino, E., Yoshida, N.: Global escape in multiparty session. In: FSTTCS 2010 (2010) (to appear), `http://www.di.unito.it/~capecchi/mpe.pdf`
12. Carbone, M., Honda, K., Yoshida, N.: Structured Communication-Centred Programming for Web Services. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 2–17. Springer, Heidelberg (2007)
13. W3C Web Services Choreography Description Language, `http://www.w3.org/2002/ws/chor/`
14. Chave, A., Arrott, M., Farcas, C., Farcas, E., Krueger, I., Meisinger, M., Orcutt, J., Vernon, F., Peach, C., Schofield, O., Kleinert, J.: Cyberinfrastructure for the US Ocean Observatories Initiative. In: Proc. IEEE OCEANS 2009. IEEE, Los Alamitos (2009)
15. Damas, L., Milner, R.: Principal type-schemes for functional programs. In: POPL, pp. 207–212 (1982)
16. Deniélou, P.-M., Yoshida, N.: Buffered communication analysis in distributed multiparty sessions. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 343–357. Springer, Heidelberg (2010) Full version, Prototype at, `http://www.doc.ic.ac.uk/~pmalo/multianalysis`
17. Deniélou, P.-M., Yoshida, N.: Dynamic multirole session types. In: POPL 2011. ACM, New York (2011) (to appear), `http://www.doc.ic.ac.uk/~malo/dynamic`
18. Desai, N., Chopra, A.K., Arrott, M., Specht, B., Singh, M.P.: Engineering foreign exchange processes via commitment protocols. In: IEEE SCC, pp. 514–521 (2007)
19. Dezani-Ciancaglini, M., Mostrous, D., Yoshida, N., Gairing, M.: Session Types for Object-Oriented Languages. In: Hu, Q. (ed.) ECOOP 2006. LNCS, vol. 4067, pp. 328–352. Springer, Heidelberg (2006)
20. Frankel, D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley, Chichester (January 2003)
21. Hoare, T.: An axiomatic basis of computer programming. CACM 12 (1969)
22. Hoare, T.: Communicating Sequential Processes. Prentice Hall, Englewood Cliffs (1985)
23. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type disciplines for structured communication-based programming. In: Hankin, C. (ed.) ESOP 1998. LNCS, vol. 1381, pp. 122–138. Springer, Heidelberg (1998)
24. Honda, K., Yoshida, N., Carbone, M.: Multiparty Asynchronous Session Types. In: POPL 2008, pp. 273–284. ACM, New York (2008)
25. Hu, R., Kouzapas, D., Pernet, O., Yoshida, N., Honda, K.: Type-safe eventful sessions in Java. In: D'Hondt, T. (ed.) ECOOP 2010. LNCS, vol. 6183, pp. 329–353. Springer, Heidelberg (2010)
26. Hu, R., Yoshida, N., Macko, M.: Session-Based Distributed Programming in Java. In: Ryan, M. (ed.) ECOOP 2008. LNCS, vol. 5142, pp. 516–541. Springer, Heidelberg (2008)
27. International Telecommunication Union. Recommendation Z.120: Message sequence chart (1996)

28. Meyer, B.: Applying "Design by Contract". Computer 25(10), 40–51 (1992)
29. Milner, R.: Theory of type polymorphism in programming languages. In: TCS (1982)
30. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes, Parts I and II. Info. & Comp. 100(1) (1992)
31. Mostrous, D., Yoshida, N., Honda, K.: Global principal typing in partially commutative asynchronous sessions. In: Castagna, G. (ed.) ESOP 2009. LNCS, vol. 5502, pp. 316–332. Springer, Heidelberg (2009)
32. Ng, N.: High performance parallel design based on session programming. Masters thesis, Department of Computing, Imperial College London (2010), `http://www.doc.ic.ac.uk/~cn06/individual-project/`
33. Nielsen, L., Yoshida, N., Honda, K.: Multiparty symmetric sumtypes. Technical Report 8, Department of Computing, Imperial College London (2009), To appear in Express'10. Apims Project at, `http://www.thelas.dk/index.php/apims`
34. OMG. Unified Modelling Language, Version 2.0 (2004)
35. Ocean Observatories Initiative (OOI), `http://www.oceanleadership.org/programs-andartnerships/ocean-observing/ooi/`
36. pi4soa homepage, `http://pi4soa.sourceforge.net/`
37. Scribble development tool site, `http://www.jboss.org/scribble`
38. Sivaramakrishnan, K.C., Nagaraj, K., Ziarek, L., Eugster, P.: Efficient session type guided distributed interaction. In: Clarke, D., Agha, G. (eds.) COORDINATION 2010. LNCS, vol. 6116, pp. 152–167. Springer, Heidelberg (2010)
39. Takeuchi, K., Honda, K., Kubo, M.: An Interaction-based Language and its Typing System. In: Halatsis, C., Philokyprou, G., Maritsas, D., Theodoridis, S. (eds.) PARLE 1994. LNCS, vol. 817, pp. 398–413. Springer, Heidelberg (1994)
40. Trivedi, A.: Hotplug in a multikernel operating system. Master's thesis, ETH Zurich (2009)
41. UNIFI. International Organization for Standardization ISO 20022 UNIversal Financial Industry message scheme (2002), `http://www.iso20022.org`
42. Welch, P., Barnes, F.: Communicating Mobile Processes: introducing occam-pi. In: Abdallah, A.E., Jones, C.B., Sanders, J.W. (eds.) CSP 2004. LNCS, vol. 3525, pp. 175–210. Springer, Heidelberg (2005)
43. Yoshida, N., Deniélou, P.-M., Bejleri, A., Hu, R.: Parameterised multiparty session types. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 128–145. Springer, Heidelberg (2010)
44. Yoshida, N., Vasconcelos, V.T., Paulino, H., Honda, K.: Session-based compilation framework for multicore programming. In: de Boer, F.S., Bonsangue, M.M., Madelaine, E. (eds.) FMCO 2008. LNCS, vol. 5751, pp. 226–246. Springer, Heidelberg (2009)

Open Government in Policy Development: From Collaborative Scenario Texts to Formal Policy Models

Maria A. Wimmer

University of Koblenz-Landau, Institute for IS Research,
Universitätsstraße 1, 56070 Koblenz, Germany
wimmer@uni-koblenz.de

Abstract. The technical capacities of service offers for e-government and e-participation have considerably progressed over the last years. Yet, the principles of good governance are still not well implemented, especially when it comes to policy development. Governments struggle to effectively apply innovative technologies in regards to providing open collaboration in policy formulation or to monitor and evaluate policy implementation. Through a recent initiative of the European Commission (EC), several research projects have been launched to address these challenges. This paper first investigates existing deficiencies in open government towards transparent policy development. Subsequently, an approach of a project funded by the EC is introduced to develop better ICT support for open collaboration in policy modeling. The approach combines existing e-participation tools, collaborative scenario generation and formal policy modeling to evaluate and explore policies via agent-based modeling (OCOPOMO - www.ocopomo.eu).

Keywords: E-Government, E-Governance, Open Collaboration, Scenario Generation, Policy Modeling.

1 From Technology-Driven E-Government to Open Government

Over the past ten years, electronic government (also called e-government or digital government) has evolved to a prevalent field of research and practice. A considerable number of definitions have emerged in this time, among which - in the European domain - the definition of the European Commission (EC) is considered as the reference of understanding in both, research and practice. The most recent definition of the EC for e-government refers to the use of information and communication technologies (ICT) by public administrations to provide better public services to citizens and businesses. According to the EC's understanding, "*effective e-government also involves rethinking organizations and processes, and changing behavior so that public services are delivered more efficiently to the people who need to use them. Implemented well, e-government enables all citizens, enterprises and organizations to carry out their business with government more easily, more quickly and at lower* *cost*".[1] Earlier definitions of the EC and other research scholars also include the effective interaction of public administrations among themselves to reduce bureaucracy and red tape as well as policy formulation and citizen participation in democratic processes. Investigations and comparison of different definitions can be found e.g. in [4], [23], [54], [55], [56].

After a first hype of e-government research and implementation in the early 2000ers, e-government had to face a severe recession around 2005, where the initiatives were blamed to be too much technology-driven and neglecting user perspectives, organisational aspects and the legal and policy dimensions in ICT developments for public administration. Subsequently, a more holistic approach involving different disciplines was acknowledged, which has been actually already claimed by the author for e-government in publications dating back to the early 2000ers (see e.g. [53], [57], [58]). Multi-disciplinary e-government research has e.g. been described in [53].

Along the discussions and evolution of innovative ICT for e-government in the past five years, topics such as citizen at focus, accessible and user-driven e-government, openness and transparency, trustworthiness, efficiency, value-generating public services, citizen participation, or open government including citizen involvement in policy making emerged from a broader understanding of e-government. The annual conference proceedings of EGOV, HICSS e-government tracks and dg.o evidence these strands of focus.

With holistic e-government development, the concept of good governance has to be referred to as well. Good governance describes the principles, approaches and guidelines for good governance and public administration to promote interaction and formation of political will with regard to societal and technological changes. Already in 2001, the EC has formulated five principles for good governance: openness, participation, accountability, effectiveness and coherence [13]. For a number of years, governance and strategic policy making were addressed separately and were not researched with the focus of using ICT. After the first downward tendency of e-government developments around 2004/2005, a high demand to focus more on the aspects of good governance in e-government arose. Two terms coined these developments: e-participation and e-governance. While e-participation concentrated on citizen participation in democratic decision making and policy formulation therewith using modern ICT[2], e-governance and public governance developments focused on organizational and efficiency aspects.

Recent trends bring together the three foci of past evolutions in research and practice of e-government, e-participation and e-governance. The OECD has published a study in 2009, where it addresses the integration of these three aspects. In the study "Focus on Citizens: Public Engagement for Better Policy and Services", it is argued that "*open and inclusive policy making offers one way to improve policy performance and meet citizens rising expectations. Public engagement in the design and delivery of public policy and services can help governments better understand people's needs,*

[1] `http://ec.europa.eu/information_society/activities/egovernment/index_en.htm` (accessed 17/10/2010).

[2] For definitions and scoping the field of e-participation, see www.demo-net.org and e.g. [35], [36], [19].

leverage a wider pool of information and resources, improve compliance, contain costs and reduce the risk of conflict and delays downstream" [43, p. 21]. Likewise, the European Commission (EC) has introduced an objective in its recent Framework Program (FP 7), which is dedicated to "ICT for governance and policy modeling"[3]. The objective specifically focuses the involvement of the general public in policy making and in strategic decision making thereby exploiting advanced ICT.

In the context of the EC FP 7, the project OCOPOMO was composed and is co-funded. OCOPOMO is the acronym for "Open COllaboration in POlicy MOdeling"[4]. The objective of the project is to define and demonstrate a new approach to policy formation that resolves crucial issues involved with prevailing approaches in policy modeling and stakeholder engagement. The innovative OCOPOMO approach is "off the mainstream" by applying a bottom-up approach to social policy modeling, combined with e-governance tools and techniques, and advanced ICT technologies. OCOPOMO will create an ICT-based environment integrating lessons and practical techniques from complexity science, agent based social simulation, foresight scenario analysis and stakeholder participation in order to formulate and monitor social policies to be adopted at several levels of government. With the open collaboration platform to policy generation, OCOPOMO enables actors of relevant target groups to better cope and master future developments and therewith meet the demands of evolving societies. Likewise, governance aspects such as transparency, trustworthiness and participation are enabled.

In the subsequent sections, current deficiencies in open government will be analyzed (section 2), the OCOPOMO project and overall approach will be introduced (section 3) and the status of relevant concepts and solutions will be investigated (section 4). Concluding remarks provide a reflection of the approach and a view on next steps in the project.

2 Current Deficiencies in Open Government

Making Europe the most competitive knowledge society is a strategic objective stressed in the Lisbon agenda [12] in the early 2000s. Since then, a number of research frameworks and strategic initiatives have been launched by the EC to spur innovative developments, application and wide-spread usage of ICT in civil society as well as the commercial and public sectors (see e.g. [11], [14], [15], [16]). While these programs and strategies have to a large extent investigated traditional ICT usage and modernization of Governments through improved online provision of public services, the areas of ICT usage in participation and especially advanced ICT support in policy making and governance were not sufficiently addressed. The most recent strategy document of the EC – "Digital Agenda for Europe 2010 – 2020"[5] – aims at advancing

[3] Objective 7.3 in call 4 of 2008, Objective 5.6 in Call 7 of 2010 (accessed 17/10/2010) – see `http://ec.europa.eu/information_society/activities/egovernment/research/fp7/index_en.htm`

[4] `http://www.ocopomo.eu/`

[5] `http://ec.europa.eu/information_society/digital-agenda/index_en.htm` (accessed 18/10/2010).

and ensuring a flourishing digital economy in Europe by 2020 [17]. This latest EC strategy addresses current deficiencies in open government and in dealing with social and economic phenomenon such as demographic change and the recent financial crisis. The “Digital Agenda for Europe” puts forward the need for actions that need to be taken urgently to get Europe on track for smart, sustainable and inclusive growth. It is expected that these actions will set the scene for the longer-term transformations that the increasingly digital economy and society will bring about. To exit the crisis and prepare the EU economy for the challenges of the next decade, Europe 2020 sets out a vision to achieve high levels of employment, a low carbon economy, productivity and social cohesion, to be implemented through concrete actions at EU and national levels. This battle for growth and jobs requires ownership at top political level and mobilization from all actors across Europe.

Especially the recent economic and financial crisis, which resulted also from our inability to predict dramatic changes in the economy and society and/or ignoring those few individuals who were warning the governments before these threats and negative trends, sheds light on an urgent need for more effective and efficient processes of governance and policy making.

When preparing for the OCOPOMO project, several deficiencies in current governance in the public sector have been unveiled, among which the following are most prominent[6]:

- inappropriate ICT support in foresights, especially in long-term policy planning,
- lack/inability of managing complexity in strategic planning and policy making in complex socioeconomic environments,
- lack of open collaboration and lack of transparency in identifying the crucial features of complex social and macroeconomic models to simulate potential alternative policies,
- ignorance of the need for e-participation and other forms of ICT-enabled efficient collaboration of communities of stakeholders relevant to the given policy area;
- lack of focus on developing, visualizing and simulating appropriate policy models to enable better management of socio-economic developments and the identification of interdependencies that result in complex social and economic relations likely to affect future developments, and
- lack of comprehensive ICT solutions to support policy modeling and simulation on the one hand, and collaboration among policy analysts and policy operators as well as wider interest groups and the general public on the other hand.

These challenges and deficiencies emerge from a long history of scientific development that resulted in the need to engage stakeholders in the design and evaluation of models of social processes, from which it was a small and natural step to use the models for purposes of social policy analysis and which itself required the engagement of the stakeholders in policy analysis. A driving force of this scientific development was the combination of two specific failures of prevailing social theory and related social modeling.

[6] As stated in the technical annex of the project contract.

The first of these failures has been the patent inability of social scientists to forecast the timing, magnitude or duration of such extreme events as the credit crisis and recession of the last two years. Despite a widespread search of the academic literature and the mass media and public enquiries via relevant newsgroups and discussion lists, the policy expert partners in OCOPOMO have not been able to identify a single, correct, model-based forecast of a turning point in either macroeconomic trade cycles or financial market indices in the whole history of economic forecasting. This problem has implications of enormous, global significance since macroeconomic and econometric models are central to the understanding of economists and the advice they give in the process of policy formation. Different models imply and inform different policies. The effects of stabilization policies and recovery plans, the magnitude of problems attendant upon protectionist policies, the need for financial support for individuals and enterprises are all assessed by the use of economic and econometric models.

The second failure is the lack of connection between scientific and informal observation of human behavior and social interaction on the one hand and, on the other, the specification of behavior by constrained optimization algorithms and the assumption that there is no social interaction in many social (and all mainstream economic) policy models. The explanation for the systematic (and possibly complete) forecasting failure emerged in parallel and in mutual ignorance of one another in two branches of the scientific literature: statistical physics and agent-based social simulation modeling.

In consequence, a new approach is necessary that builds on evidence-based user-driven development of policy texts, which informs a formal policy model to elicit potential interdependencies, and which enables a simulation model as correct and precise as possible. The development of the user-generated scenario models and the formal simulation models needs to be iterative and needs to enables different relevant stakeholders to interact and contribute to the evidence-based policy texts.

3 OCOPOMO: A New Approach to Integrate Open Collaboration in Policy Development

3.1 The Project

To address current deficiencies in open government and to cope with the challenges and shortcomings of currently existing fragmented approaches, OCOPOMO provides an innovative "off the mainstream" bottom-up approach to social policy modeling, combined with e-governance tools and techniques and advanced ICT technologies. It therewith integrates concepts, techniques and tools from different disciplinary fields in order to formulate and monitor social policies. Focus is put on the long-term strategic planning for governments and policy operators (those responsible for preparing and developing the strategic decisions to be made in parliaments and or respective government bodies).

OCOPOMO aims at providing an integrated ICT toolbox with proper mechanisms for open collaboration in policy modeling, including collaborative support for scenario based policy development. It will enable actors of all target groups at different levels of government across Europe "*to master and shape future developments so that the demands of its society and economy are met*"[7]. Policy issues which are high on the European political agenda serve as test-beds to evaluate and test the OCOPOMO approach. The policy cases selected for this purpose are renewable energy and management of structural funds. With the two test cases, OCOPOMO demonstrates that, with appropriate ICT, the integration of formal policy modeling, scenario generation and open and widespread collaboration is not only possible but can come to be seen as essential at all levels of policy formation - whether local, regional, national or global.

The overall concept of OCOPOMO is shown in Fig. 1. Through open collaboration (e-participation features), stakeholders develop a set of scenarios for the policy cases (1). Based on the understanding of each policy case and the most wanted scenario elements, policy experts generate a common macroeconomic model (2) and targeted individual agent-based policy models for the pilot cases (3). These formal policy models are simulated (4) and visualized to enable stakeholders (5a) and policy modeling experts (5b) to validate and evaluate the simulated policy models. In several iterations of scenario and model development, the policy models are refined.

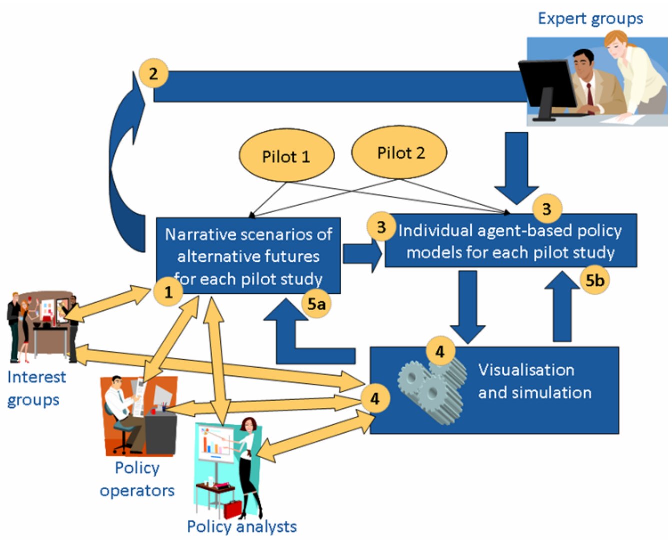

Expert groups
2
Pilot 1
Pilot 2
3
Narrative scenarios of alternative futures for each pilot study
3
Individual agent-based policy models for each pilot study
1
5a
5b
4
Visualisation and simulation
4
Interest groups
Policy operators
Policy analysts

Fig. 1. Overall concept for open collaboration in policy modeling in the OCOPOMO project

Through open and widespread collaboration via the ICT toolbox, scenario generation and formal policy modeling, the policy experts as well as wider stakeholder groups are supported in strategic decision making and policy formation. The OCOPOMO approach thereby provides a more suitable policy approach and engages the stakeholder in different stages of the policy formulation.

[7] `http://cordis.europa.eu/fp7/ict/` (accessed 31st October 2010).

To achieve the core objectives of OCOPOMO, the project will generate the following outcomes:

- Two policy analyses at regional level within Member States of the European Union, which will be based on, and enriched with different sets of user-generated scenarios that reflect particular perspectives of relevant stakeholder views in each policy case. The pilot policy cases cover different political, cultural and geographical environments. The policy issues to be addressed are all sensitive to the current financial and economic crisis and economic development policies. The policy analyses will be based on both, formal simulation models and narrative scenarios.
- A general model of macroeconomic relations constrained as far as possible by data produced at national and European level. The model will have properties that are well justified by evidence in every case, some of which are known from experience to yield the unpredictable extreme events and statistical properties of outputs from the models that are incompatible with statistical forecasting techniques.
- Narrative scenario analyses that inform the formal policy models in a way to produce policy analyses with the precision and clarity of formal models and also the rich contextual and imaginative content of verbal narratives of relevant stakeholders in the respective policy cases.
- A model of macroeconomic complexity enriched with the regional policy models, which will ensure that the regional models generate a range of surprising results that can be analyzed both formally from model output and informally by means of scenario exercises, online forum, and the like.
- An integrated ICT solution, which will support the engagement of core, participating stakeholders and also open engagement by stakeholders who are not partners in the project but who have an interest or expertise in the policy issues.

The ICT solution will provide a collaborative environment for an integrated process of evidence-based user-generated scenario development and formal policy modeling, which will produce formal model-driven scenarios by means of simulation experiments. The target users of the OCOPOMO toolbox are on one hand policy analysts and policy operators and, on the other hand, special interest groups and to some extent the wider general public. Hence, the traditional approach of (expert) top-down policy modeling is counterbalanced and expanded with

1. innovative ground-up participation in (narrative) scenario-building and
2. an iterative process of identifying the parameters and features of policy models from the narrative scenarios, designing and simulating the policy models (including outputs of formal scenarios) and refining them iteratively;
3. open collaboration of policy analysts, policy operators and wider interest groups (representatives of specific unions, chambers, etc., and the general public).

3.2 The Approach in More Detail

Using the numbered items depicted in Fig. 1, policy operators (decision makers in governments and politics), policy analysts and interest groups (specific stakeholder groups as well as general public) collaborate in the process of scenario development.

Thereby, they depict alternative narrative descriptions of a policy area, which demands strategic decisions (1). Local development policies embrace a wide range of issues, e.g. Natural resources (improvements to water resources, soil and coastline protection, upgrading of natural areas, waste processing and energy management (especially renewable resources), Cultural resources (Enhancement of the region's cultural resources as a factor contributing to its economic and social development.), employment and training policies. The results of this process step are a number of alternative scenarios for each policy domain (i.e. for each pilot).

Based on these narrative scenarios, two types of model will be developed. The first type (2 in Fig. 1) is a common macroeconomic agent-based simulation model that produces outputs relevant to the policy concerns of each of the regional pilot models. Because the macroeconomic model has the properties of complexity models more generally, we expect it to produce surprises for the regional stakeholders or, at least, to encourage them to plan for unwanted extreme events such as a collapse in private investment or consumption or unexpected changes in rates of inflation (positive or negative). This common macroeconomic model will be integrated into each of the regional pilot models to simulate the wider economic environment in which the regional policies are to be implemented. The regional pilot models (3 in Fig. 1) will be developed specifically to reflect the concerns, objectives and perspectives of the local stakeholders.

Subsequently, the policy models can be simulated and visualized (4 in Fig. 1). The interest groups, local policy analysts and policy operators simulate and evaluate the policy models based on the scenarios developed in step 1. The aim of this step is to assess and evaluate the policy models and the scenarios developed. The result of this step is a collection of revisions and modifications for the alternative scenarios, the parameters identified as those being crucial in the policy domain, and the individual policy models.

These results of step four are fed back to earlier steps – depending on what revisions are requested, and where – in order to revise: a) the alternative scenarios developed which starts the process again at step 1 (5a in Fig. 1) or b) the individual policy models at step 3 (iteration indicated with (5b) in Fig. 1). Throughout the execution of the OCOPOMO project, two iterations are scheduled for this process. In practice, this process is likely to be ongoing and incremental with many partial revisions.

Finally, the resulting narrative scenarios and policy models shall help policy operators to make their decisions on the basis of (i) better quality of policy analysis results available and (ii) a consultative process that will have taken place with key stakeholders of the policy domain throughout the policy development.

Likewise, interest groups affected by such decisions have better knowledge and understanding of the decisions to be taken by policy makers, as they have actively contributed their views, concerns and understanding to the policy process.

3.3 Overall Architecture for the ICT Toolbox

Current ICT tools in e-participation focus on participation with either advanced means of search, interaction or argument visualization. In this context, the great majority of e-participation platforms and social networks based on Web 2.0 technologies serve as examples. Other tools – not necessarily with widespread usage by the general public - provide ICT support in policy modeling and simulation.

In OCOPOMO we aim at bringing together both strands of tools and technologies available. Fig. 2 shows the overall architecture of the OCOPOMO ICT solution. The OCOPOMO platform will allow on one hand open collaboration in scenario development (step 1 of Figure 1) through the use of advanced participation and collaboration features. Such tools provide means for structuring joint scenario development, discussion, assessment and rating features of the key parameters to be identified. On the other hand, tools for policy modeling, policy visualization and simulation will be integrated in order to support the different target groups in performing the steps 2, 3 and 4 of the procedural design shown in Figure 1.

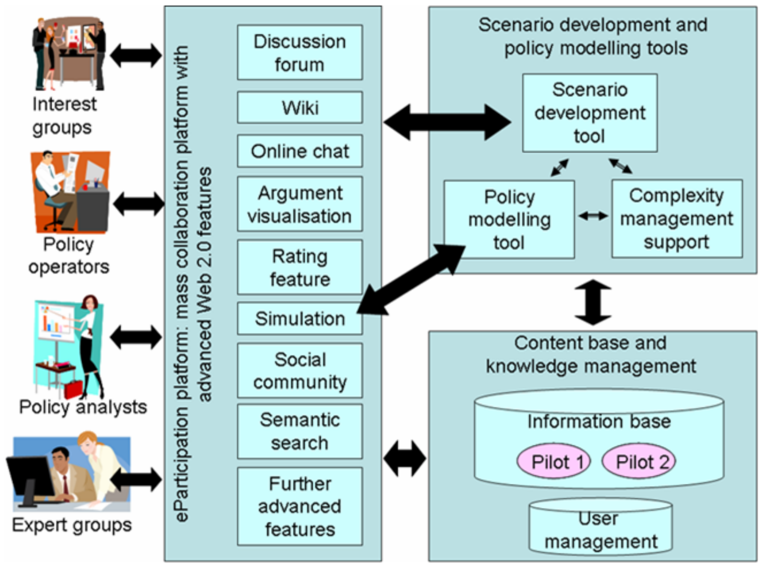

Interest groups
Policy operators
Policy analysts
Expert groups
eParticipation platform: mass collaboration platform with advanced Web 2.0 features
Discussion forum
Wiki
Online chat
Argument visualisation
Rating feature
Simulation
Social community
Semantic search
Further advanced features
Scenario development and policy modelling tools
Scenario development tool
Policy modelling tool
Complexity management support
Content base and knowledge management
Information base
Pilot 1
Pilot 2
User management

Fig. 2. OCOPOMO's ICT toolbox supporting open collaboration of stakeholders in scenario-based policy development, as well as policy experts in policy analysis, modeling, simulation and complexity management

To achieve the objectives in OCOPOMO, mostly existing tools will be integrated. Hence, the next section provides an overview of relevant tools and developments.

4 Current Status of Related Developments

OCOPOMO integrates a number of currently existing concepts and ICT solutions, which exist for particular purposes. Among the research and development topics of high relevance in OCOPOMO, the following are briefly introduced in this section: policy modeling (including in regards to e-governance), complexity in policy modeling and public governance, social policies in relation to e-governance, scenario-based foresights, e-governance and e-participation, and ICT solutions for integrating user-generated scenarios with formal policy modeling therewith enabling open collaboration among relevant stakeholders. OCOPOMO aims at integrating these solutions to a comprehensive innovative toolbox.

Agent based *policy modeling* is an application of agent based social simulation. The field can be divided into two classes: models that are designed and implemented bottom-up on the basis of evidence obtained from stakeholders and other domain experts, and models that are designed and implemented top-down on the basis of some prior social (frequently economic) theory. Top-down, theory driven models are largely used in economics to suggest policy implications. Different approaches to macroeconomic modeling exist, yet many of them lack representation of individual behavior based directly or indirectly on evidence of how individuals behave or how they interact (cf. e.g. [9], [24], [26], [49]). To model the overall economy at the macro-level, the model of Nobel Prize winner Larry Klein is one approach [33]. It essentially builds on the Keynesian system of macro-theoretical relationships and it adds fitting curves (planes) through empirical macro-data using econometric (statistical) techniques to estimate parameters for these theoretical relationships. Another more recent approach is based upon micro-theoretical relationships. It posits that the economy can be understood by reference to a single so-called ‘representative agent’, who takes decisions to maximize his/her utility over an infinite time horizon (cf. e.g. [52]). In OCOPOMO, a bottom-up evidence-based approach will be used. It will therewith use multi-agent concepts as e.g. put forward by [38], [39]. This way, complexity management concepts are engaged alike.

Complexity represents an interdisciplinary approach in science, which focuses on explaining highly complex phenomena in terms of simple rules. The essence of this approach is well captured by the statement of P.W. Andersen [1] that “more is different”. The discovery that complex properties may emerge from simple elements interacting with each other in a simple way is one of the most important discoveries of modern science (e.g. [27], [31], [48]). Even if the system’s elements are relatively simple, nonlinearity in their interactions may lead to highly complex dynamic behavior, such as self-organization, and pattern formation (cf. [6]; [25]; [31], [32]; [44]). For example, simple rules of social influence among individuals lead to the emergence of complex patterns of public opinion [42]. Complex properties arise in the process of self-organization. According to principles of "dynamical minimalism" [41] the goal of complexity inspired science is to unveil the simple rules governing interaction of system's elements and to show how interactions of these rules in time can produce the complexity observed at the system level. Many models for complex systems with phenomena of economics have been proposed (cf. e.g. [3], [10], [34], [37], [46]). Moss et al. [39] explicitly introduce purely social factors into a model of self-organized criticality and therewith demonstrated that models attributing self-organized criticality to the interaction between rationally reasoning agents has the same statistical signatures as real markets for a wide range of goods and services [38].

The impact of *social* (including economic) *policies* is frequently conditioned by wider economic considerations. Unpredictability is a key aspect of events that could occur during the course of any but the shortest-term policy implementations. The kinds of indicators that will be used to assess the effectiveness and feasibility of policies and the responses to unexpected events will have to be considered by stakeholders engaging in policy analyses that take into account the outputs from models of social complexity.

In these cases, the responses can involve actions and considerations that are identified in new narrative scenarios exemplified in the scenario development approach used in the eGovRTD2020 project [8]. Agents in agent-based models can learn to combine elementary actions in novel ways to formulate policies and plans but cannot identify novel actions. Consequently, the additional actions and considerations identified by stakeholders in response either to unexpected real events or simulated outcomes will then have to be incorporated into further versions of the policy models. Complexity evidently brings to the fore the complementarity between narrative scenario exercises and formal simulation modeling, which will be combined in OCOPOMO throughout an iterative approach.

The role of *scenario studies* is commonly taken to be the definition and exploration of policy vies that are possible without attaching any subjective or statistical probabilities that any one of them will actually occur. The scenarios are narratives couched in the language of participating stakeholders and domain experts [7]. Scenario-building is a technique of future research that aims at generating different perspectives of the future to gain more insight into possible opportunities and threats. This technique allows better and more effective exploration of alternative trajectories of a certain domain beyond short-term forecasting. Different approaches exist [20]. Scenarios may e.g. start from an actual problem which is perceived as disappointing by a large part of the population and which must urgently be solved. Additionally there are several (sometimes controversial) scientific and/or political approaches to solve the problem, which can be documented in alternative scenario descriptions. Although the future is complex and therefore largely unpredictable, forming clear perspectives and cognitive frameworks for considering alternative trajectories is extremely valuable. Foresight scenarios have been developed for discussions of policies and strategies relating to future trajectories that are subject to uncertainty. Prime examples are environmental issues (e.g. IPCC scenarios [40], research and development strategies (e.g. eGovRTD2020 [8]). More detailed argumentation on the use of scenario methods and an exemplary application are available in [28] and [29].

As already mentioned in the introduction, the EC has formulated five principles for *good governance*. Besides, the UN ESCAP [51] defined eight major characteristics for good governance: participatory, consensus oriented, accountable, transparent, responsive, effective and efficient, equitable and inclusive and in conformity with the law. According to ESCAP, governance is "*the process of decision-making and the process by which decisions are implemented, an analysis of governance focuses on the formal and informal actors involved in decision-making and implementing the decisions made and the formal and informal structures that have been set in place to arrive at and implement the decision*" [51]. Governance thereby fosters the managerial, organizational and policy aspects of steering States and democracies. Good governance characteristics differ depending on country, culture or chosen point of view. Overall, governance aims at minimizing corruption and taking the views of minorities into account in decision-making thereby being responsive to the present and future needs of society. A crucial aspect of good governance to be addressed in OCOPOMO is the involvement of formal and informal actors in policy making and in strategic decision making, as a more systematic dialogue among relevant stakeholders thereby improving in particular the dialogue with non-governmental and private actors when developing policy proposals with wide-ranging consequences. With the involvement of different affected stakeholders and experts in an online open collaboration (using e-participation tools and techniques), an approach to ensure policy coherence is given. Combined with e-participation facilities, the OCOPOMO approach represents an innovative means for involved stakeholders and for a wider interest group to take part in the opinion-making process up to the point of decision-making with electronic systems. Therewith, public responsiveness and public satisfaction can be improved.

The *ICT solutions for policy modeling, scenario-building and the open collaboration platform* of OCOPOMO embark on leading edge tools and technologies in several fields (cf. Fig. 2). Key elements to be adapted, integrated and deployed for the OCOPOMO ICT governance toolbox are:

- Agent-based modeling tools supporting formal model development and policy simulation (cf. e.g. [18]), especially multi-agent systems. The advantages of MAS such as heterogeneity, modularity, flexibility and robustness against failures seem especially appropriate for building complex systems [30].[8]
- Content management systems and knowledge systems for managing the sources and various contents of the toolbox to provide efficient management of a rich set of document and content services that address a wide range of different content types (for a wider discussion of CMS in e-participation see e.g. [47]). Examples of current open source CMS systems are: Alfresco, Apache Lenya, Ariadne, AWF-CMS, Contenido, Daisy, Drupal, E107, Jackrabbit, Plone, Rainbow, Sitellite, SPINE, Typo3, WebGUI, Xaraya, Xorio, etc.
- E-participation platforms with advanced web 2.0 features, argument visualization, etc. enabling participation of wider stakeholder groups. Current e-participation platforms extensively base on web 2.0 features and social software such as discussion fora, wikis, social networks, blogs, chats, podcasts, web services, Natural language processing tools, GIS tools, etc. Extensive reports on e-participation tools and technologies are available from DEMO-net reports [19], [2], [22], [45] and in [47].

5 Concluding Remarks

In this contribution, current deficiencies in open government have been investigated. This included a review of developments in the field of e-government. As the European Commission provides co-funding of research to strengthen a current policy priority on “ICT for governance and policy modeling”, the OCOPOMO project has been introduced, which forms an innovative approach to realize open government. OCOPOMO bases on a combined approach of user-driven scenario-generation and formal policy modeling therewith facilitating participation of stakeholders in the policy formation process.

Successful integration of policy modeling and scenario analysis for use by stakeholders and policy operators has not, as far as we know, previously been attempted. Certainly, the design, implementation and running of such models has informed

[8] Popular software frameworks and toolkits for developing multi-agent systems today are JADE (`http://jade.tilab.com/`) and Repast (`http://repast.sourceforge.net`).

scenario analysis and role playing games and therefore influenced stakeholders indirectly. To achieve the direct engagement of stakeholders with policy modeling and to use that engagement in the development of complementary scenarios adds significant progress beyond the state of the art in policy modeling - not least by demonstrating a whole new approach to the use of models and scenarios in policy formation.

This achievement will itself will require significant innovations in ICT as well in terms of integration of current tools developed for specific purpose. The biggest challenge of OCOPOMO will be to ensure traceability of results from the evidence-based and user generated scenario models to the formal policy models and back. This is a topic currently under investigation and conceptualization. Studies of qualitative data analysis, text analysis and tracing via tagging are ongoing. A concept has been drafted. It will be subject to future scientific publications from the project activities.

The benefits and expected added value of OCOPOMO are, on the one hand, the implementation and proof of concept of an integrated ICT toolbox to support complex socio-economic policy making, ready-made to be deployed in similar policy domains without further large efforts of implementation. On the other hand, governments and policy operators benefit from the OCOPOMO solution by being able to involve stakeholders and have a toolbox to master complexity better in developing their strategic policies through an integrated e-governance toolbox. The innovative approach of facilitating stakeholder engagement in complex policy making domains adds value to them by combining narrative scenario development and advanced agent-based social simulation in strategic policy areas. Through the policy models, the methodological approach of combining scenario-based stakeholder engagement and formal policy modeling provides policy operators with better results in foresight and impact assessments of alternative policies. Hence, OCOPOMO adds considerable value in implementing current objectives of "ICT for governance and policy modeling" of the European Union.

Acknowledgement. OCOPOMO is co-funded by the European Commission under the 7th Framework Program. This publication reflects the view only of the author and the project consortium, and the Commission cannot be held responsible for any use which may be made of the information contained therein. This paper reflects parts of the OCOPOMO project proposal. Herewith the author expresses special thanks to her colleague Prof. Scott Moss for his invaluable contributions and discussions to shaping the project scope and proposal of OCOPOMO. Thanks to the project partners to shape the project.

References

1. Anderson, P.W., Arrow, K., Pines, D. (eds.): The Economy as an Evolving Complex System. Addison-Wesley Longman, Redwood (1988)
2. Apostolou, D., Babic, F., Bafoutsou, G., Butka, P., Dioudis, S., Mach, M., Macintosh, A., Gordon, T., Halaris, C., Kafentzis, K., Mentzas, G., Paralic, M., Paralic, J., Renton, A., Rosendahl, A., Sabol, T., Schneider, C., Thorleifsdottir, A., Wimmer, M.: D5.2 - eParticipation: The potential of new and emerging technologies (2007) DEMO-net consortium, `http://www.demo-net.org`

3. Arthur, W.B.: Complexity and the Economy. In: Colander, D. (ed.) The Complexity Vision and the Teaching of Economics. Edward Elgar (2000)
4. Barzilai-Nahon, K., Scholl, H.J.: Siblings of a Different Kind: E-Government and E-Commerce. In: Wimmer, M., Scholl, H.J., Janssen, M., Chappelet, J.-L. (eds.) EGOV 2010. LNCS, vol. 6228, pp. 25–37. Springer, Heidelberg (2010)
5. Botterman, M., Millard, J., Horlings, E., van Oranje, C., van Deelen, M., Pedersen, K.: Value for citizens - A vision of public governance in 2020. European Commission (2008), http://ec.europa.eu/information_society/activities/egovernment/studies/docs/final_report_web.pdf (accessed 17/10/2010)
6. Camazine, S.: Self-organizing systems. In: Nadel, L. (ed.) Encyclopedia of Cognitive Science, pp. 1059–1062. Elsevier, New York (2003)
7. Carroll, J.M.: Scenario-Based Design: Envisioning Work and Technology in System Development. Wiley, Chichester (1995)
8. Codagnone, C., Wimmer, M.A. (eds.): Roadmapping eGovernment Research: Visions and Measures towards Innovative Governments in 2020. MY Print snc di Guerinoni Marco & C, Clusone (2007)
9. Cortelezzi, F., Villani, G.: Valuation of R&D Sequential Exchange Options Using Monte Carlo Approach. Journal of Computational Economics 33(3), 209–236 (2009)
10. Epstein, J.M.: Generative Social Science: Studies in Agent-Based Computational Modeling. Princeton University Press, New York (2006)
11. eTen, eTen Program: support for trans-European telecommunications networks. Trans-European Telecommunications Networks (2007), http://europa.eu/legislation_summaries/information_society/l24226e_en.htm (accessed 18/10/2010)
12. European Commission. Lisbon European Council 23 And 24 March 2000 (2000), http://www.europarl.europa.eu/summits/lis1_en.htm (accessed 18/10/2010)
13. European Commission, European Governance, A white paper, COM (2001) 428 final, Brussels (25.7.2001), http://eur-lex.europa.eu/LexUriServ/site/en/com/2001/com2001_0428en01.pdf (accessed 18/10/2010)
14. European Commission. eEurope 2005. An information society for all: An Action Plan to be presented in view of the Sevilla European Council, COM (2002) 263 final. Brussel (2002), http://ec.europa.eu/information_society/eeurope/2002/news_library/documents/eeurope2005/eeurope2005_en.pdf (accessed 18/10/2010)
15. European Commission. Establishing a Competitiveness and Innovation Framework Programme (2007-2013), COM (2005) 121 final. Brussels (2005), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2005:0121:FIN:EN:PDF (accessed 18/10/2010)
16. European Commission. i2010 - A European Information Society for growth and employment, COM (2005) 229 final. Brussels (2005), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2005:0229:FIN:EN:PDF (accessed 18/10/2010)
17. European Commission: A Digital Agenda for Europe. Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions, COM (2010) 245 final/2, Brussels (26.8.2010), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0245:FIN:EN:PDF (accessed 18/10/2010)

18. Fletcher, M., Vrba, P.: A Brace of Agent Simulation Scenarios. In: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS 2006), pp. 169–176 (2006)
19. Fraser, C., Liotas, N., Lippa, B., Mach, M., Macintosh, A., Marzano, F., Mentzas, G., Rosendahl, A., Sabol, T., Tambouris, E., Tarabanis, K., Thorleifsdottir, A., Westholm, H., Wimmer, M.: D5.1 - Report on current ICTs to enable Participation. DEMO-net consortium (2006), `http://www.demo-net.org` -> Results (accessed 17/10/2010)
20. Gausemeier, J., Fink, A., Schlake, O.: Szenario-Management: Planen und Führen mit Szenarien. München, Hanser (1995)
21. Gell-Mann, M.: The quark and the jaguar: Adventure in the simple and the complex. Freeman, New York (1994)
22. Gkarafli, M., Papadopoulos, A., Tambouris, E., Tarabanis, K.: D14.3c: The role of Web 2.0 technologies in eParticipation, DEMO-net consortium (2007), `http://www.demo-net.org`
23. Grönlund, A.: Ten Years of eGovernment: The 'End of History' and New Beginning. In: Wimmer, M.A., Chappelet, J.-L., Janssen, M., Scholl, H.J. (eds.) EGOV 2010. LNCS, vol. 6228, pp. 13–24. Springer, Heidelberg (2010)
24. Guastaroba, G., Mansini, R., Grazia Speranza, M.: Models and Simulations for Portfolio Rebalancing. Journal of Computational Economics 33(3), 237–262 (2009)
25. Haken, H.: Synergetics. Springer, Berlin (1978)
26. He, L.T., Hu, C.: Impacts of Interval Computing on Stock Market Variability Forecasting. Journal of Computational Economics 33(3), 263–276 (2009)
27. Holland, J.H.: Emergence: From chaos to order. Addison-Wesley, Reading (1995)
28. Janssen, M., van der Duin, P., Wimmer, M.A.: Framework and Methodology: Methodology for scenario building. In: 8, pp. 23–28 (2007)
29. Janssen, M., Wimmer, M.A., Bicking, M., Wagenaar, R.W.: Scenarios of governments in 2020. In: 8, pp. 55–84 (2007)
30. Jennings, N.R., Bussman, S.: Agent-Based Control Systems: Why are They Suited to Engineering Complex Systems? IEEE Control Systems Magazine 23(3), 61–73 (2003)
31. Johnson, S.: Emergence: The connected lives of ants, brains, cities and software. Scribner, New York (2001)
32. Kelso, J.A.S.: Dynamic patterns: The self-organization of brain and behavior. MIT Press, Cambridge (1995)
33. Klein, L.R.: The use of econometric models for policy purposes'. Econometrica 15 (1947)
34. Levy, M., Levy, H., Solomon, S.: Microscopic Simulation of Financial Markets: From Investor Behavior to Market phenomena. Academic Press, New York (2000)
35. Macintosh, A.: Characterizing e-participation in policy-making. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS-37), Track 5, vol. 5, p. 50117a. IEEE Computer Society Press, Washington (2004), `http://doi.ieeecomputersociety.org/10.1109/HICSS.2004.1265300`
36. Macintosh, A. (ed.): The Initial DEMO-net Landscape. Deliverable D4.1, DEMO-net Consortium (2006), `http://www.demo-net.org`-> Results
37. Miller, J.H., Page, S.E.: Complex Adaptive Systems: An Introduction to Computational Models of Social Life. Princeton University Press, New York (2006)
38. Moss, S.: Competition in Internal Markets: Statistical Signatures and Critical Densities. CPM Report Number 01-79, Manchester Metropolitan University, UK (2001)

39. Moss, S., Edmonds, B., Wallis, S.: The Power Law and Critical Density in Large Multi-Agent Systems. CPM Report Number 00-71, Manchester Metropolitan University, UK (2000)
40. Nakicenovic, N., Swart, R.: Emissions Scenarios: Special Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge (2000)
41. Nowak, A.: Personality and Social Psychology Review 8(2), 183–192 (2004)
42. Nowak, A., Szamrej, J., Latane', B.: From private attitude to public opinion: A dynamic theory of social impact. Psychological Review 97, 362–376 (1990)
43. OECD: Focus on Citizens: Public Engagement for Better Policy and Services, OECD Studies on Public Engagement. OECD Publishing (2009) doi: 10.1787/9789264048874-en
44. Prigogine, I., Stengers, I.: Order out of chaos. Bantam, Toronto (1984)
45. Rose, J., Sæbø, O., Nyvang, T., Sanford, C.: D14.3a: The role of Social networking software in eParticipation. DEMO-net consortium (2007), `http://www.demo-net.org`
46. Rosser, J.B.: On the Complexities of Complex Economic Dynamics. The Journal of Economic Perspectives 13, 169–192 (1999)
47. Scherer, S., Wimmer, M.A., Schneider, C.: Investigating Information and Knowledge Management (IKM) in eDeliberation. In: Cunningham, P., Cunningham, M. (eds.) Collaboration and the Knowledge Economy: Issues, Applications, Case Studies, pp. 270–277. IOS Press, Amsterdam (2008)
48. Stephan, A.: Emergence. In: Nadel, L. (ed.) Encyclopedia of Cognitive Science, pp. 1108–1115. Nature Publishing Group, London (2003)
49. Strid, I., Walentin, K.: Block Kalman: Filtering for Large-Scale DSGE Models. Journal of Computational Economics 33(3), 277–304 (2009)
50. Tobias, R., Hofmann, C.: Evaluation of free Java-libraries for social-scientific agent based simulation. Journal of Artificial Societies and Social Simulation 7(1), 6 (2004), `http://jasss.soc.surrey.ac.uk/7/1/6.html`
51. United Nations Economic and Social Commission for Asia and the Pacific (ESCAP): What is good governance?
52. Wellens, T., Kuś, M.: Separable approximation for mixed states of composite quantum systems. Phys. Rev. A 64, 052302 (2001)
53. Wimmer, M.A.: Integrated service modeling for online one-stop Government. EM - Electronic Markets, Special Issue on e-Government 12(3), 1–8 (2002)
54. Wimmer, M.A.: The Role of Research in Successful E-Government Implementation. In: Zechner, A. (ed.) E-Government Guide Germany. Strategies, Solutions and Efficiency, Stuttgart, pp. 79–87. Fraunhofer IRB Verlag (2007)
55. Wimmer, M.A.: Introduction. In: [8], pp. 1–9
56. Wimmer, M.A., Codagnone, C.: Framework and Methodology: Definitions for eGovernment. In: [8], pp. 11–12
57. Wimmer, M.A., Traunmüller, R.: Integration - The Next Challenge in e-Government. In: Far, B.H., Shafazand, M.H., Takizawa, M., Wagner, R. (eds.) EurAsia-ICT 2002 - Advances in Information and Communication Technology, pp. 213–218, Book series # 161. Austrian Computer Society (2002)
58. Wimmer, M.A., von Bredow, B.: A Holistic Approach for Providing Security Solutions in e-Government. In: Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35) at Big Island of Hawaii (2002)

Linear Process Algebra

Vaughan Pratt

Stanford University, Stanford CA 94305-9045, USA
pratt@cs.stanford.edu

Abstract. A linear process is a system of events and states related by an inner product, on which are defined the behaviorally motivated operations of tensor product or orthocurrence, sum or concurrence, sequence, and choice. Linear process algebra or LPA is the theory of this framework. LPA resembles Girard's linear logic with the differences attributable to its focus on behavior instead of proof. As with MLL the multiplicative part can be construed via the Curry-Howard isomorphism as an enrichment of Boolean algebra. The additives cater for independent concurrency or parallel play. The traditional sequential operations of sequence and choice exploit process-specific state information catering for notions of transition and cancellation.[1]

Keywords: concurrency, event, state, duality, linear logic, Curry-Howard.

1 Background

Computation itself, as distinct from its infrastructure (operating systems, programming languages, etc.) and applications (graphics, robotics, databases, etc.), has two main aspects, algorithmic and logical. The algorithmic aspect serves programmers by providing techniques for the design and analysis of programs. The logical aspect serves language designers, compiler writers, documentation writers, and program verification by proposing suitable concepts for the operations and constants of a language (abstract syntax), giving them meanings and names (semantics and concrete syntax), showing how to reason about them (logic), and studying their structure (abstract algebra).

Originally computation was performed on a single computer under the control of a central processing unit. Parallel and distributed computing emerged from the infrastructure with the advent of multiprocessors and networking, enhancing the applications at the expense of complicating both the algorithmic and formal aspects of computation. This paper focuses on the latter.

Formal methods divide broadly into logical and algebraic. On the logical side we find Amir Pnueli's temporal logic [45], which speaks of a single universal process from the point of view of a neutral observer. Pnueli has called this

[1] More recent follow-up remarks and expansions on this paper may be found at http:boole.stanford.edu/pub/LPA

kind of specification *endogenous* to distinguish it from *exogenous* modal logics of programs such as dynamic logic [32, 46].

In the algebraic approach process calculi provide one (but certainly not the only) framework. The most prominent of the early such calculi are Hoare's Communicating Sequential Processes (CSP) [7, 33], Milner's Calculus of Communicating Systems (CCS) [38], and Bergstra and Klop's Algebra of Communicating Processes (ACP) [2, 5, 6].

Each of these calculi imputes a certain nature to the process concept. Their differences raise the question of whether they are theories of essentially different entities or are merely refinements of a common core conception differing only in their emphases on secondary aspects.

A similar question arose millennia ago about the core of geometry, and was answered in turn by Euclidean geometry which postulated lines and circles as primitive concepts, cartesian geometry which expressed them as $ax + by + c = 0$ and $(x - x_0)^2 + (y - y_0)^2 = r^2$, and linear algebra as a source of more abstract spaces within which to conduct cartesian geometry and much else besides. Each in turn shed light on its predecessor. Yet all of them can be seen to be about Euler's notion of affine space as a common framework, with Euclidean geometry adding notions of metric and angle, linear algebra adding an arbitrary origin, and cartesian geometry doing both via the basis imputed by its coordinate frame.

Both space and computation can be understood either denotationally—what they are—or operationally—how to construct things. Euclid's account of Euclidean space was operational in that a good number of his postulates and propositions promised constructions in space rather than properties of space. Linear algebra on the other hand is founded on a denotational framework, with matrix inversion M^{-1} for example being defined denotationally as a solution in N to $MN = I$ before addressing questions of existence, uniqueness, and construction. The modern conception of Euclidean space is entirely denotational, making Euclid's operational treatment of the two-dimensional case of his eponymous space seem somehow nonmathematical to the modern reader, its trend-setting logical formulation notwithstanding.

Concurrency has likewise had both operational and denotational accounts. Among the former, perhaps the best known are Petri nets [42] and Plotkin's Structured Operational Semantics (SOS) [44]. The considerable popularity of both can be taken to mean that concurrency is best treated operationally, or that its denotational treatment is problematic, or both.

It cannot however be taken to mean that no one has tried. The denotational semantics of concurrency can be considered to have begun with the idea of sequential processes as sets of computation traces, by analogy with formal languages as sets of strings, with concurrency introduced via the shuffle operation [15, 68]. However this semantics captures neither branching (timing of nondeterminism) nor independence (determinism of nontiming), both of which can be seen as a deficiency not of the operations but of sets of traces themselves. In that model all decisions in a computation are reduced to the choice of a single trace, ignoring both the order in which the decisions were made and their timing relative to other events. Independence of a and b on the other hand is expressed as the choice $ab+ba$, introducing both order and choice when neither are relevant to independence.

Branching time was first formalized denotationally by Milner's synchronization trees [38]. The implied distinction was formalized by Park in terms of a relation of bisimilarity [41] expressing lock-step equivalence. Bisimilarity is a more refined equivalence of processes than mere equality of two sets of traces, which is too coarse to distinguish $a(b+c)$ from $ab+ac$. Subsequently a hierarchy of congruences intermediate between trace equivalence and bisimilarity emerged: Figure 1 of [73] partially orders the 11 semantics by their relative positions in the linear time/branching time spectrum.

Independence, or "true concurrency," was formalized denotationally early on by Greif [28] in terms of events partially ordered by time as a semantics for Hewitt's actor framework, and later by Mazurkiewicz via traces quotiented by an equivalence relation of independence on the alphabet Σ extended to a congruence on the free monoid Σ^* [37]. Yet later Grabowski [27] and Pratt [17, 48] further abstracted Mazurkiewicz traces with a notion of partially ordered multiset or pomset. In the latter two models choice was expressed by defining a process to be a set of traces or pomsets.

All these semantics of independence were explicitly or implicitly based on the notion of multiset over an alphabet of symbols as a set labeled with symbols. Identifying unlabeled strings with ordinals, a string can be defined as a labeled ordinal, with the length of a string being the underlying ordinal. Pomsets generalize labeled ordinals to labeled posets.

Nielsen, Plotkin and Winskel's notion of event structure $(A, \leq, \#)$ [39, 76–78] formalizes independence as for pomsets, namely with a partial order $\leq$, with two differences. First it conflates the two-level set-of-pomsets approach to the one level used in treatments of branching time by expressing choice in terms of an irreflexive symmetric conflict relation $a\#b$ on events satisfying $a\#b \wedge b \leq c \Rightarrow a\#c$. Conflict creates the choice of which of the conflicting events *not* to perform, and is the only kind of choice expressible by event structures. Second the unlabeled event structures are taken to be the primary object of study, which is analogous to studying strings over a one-letter alphabet, i.e. ordinals, instead of general strings. Unlabeled event structures are already interesting enough in their own right without labels.

But while unlabeled event structures can easily represent $a||b$, a and b acting independently, it is unclear how they can distinguish it from $ab + ba$, a and b acting in either order. Gaifman and Pratt [14] addressed this distinction with a notion of prosset $(E, \leq, <)$ with a reflexive weak and irreflexive strong partial order satisfying $a < b \;\Rightarrow\; a \leq b$, with a process defined as a set of prossets as for traces and pomsets. Both relations order events by time, with the difference being that only $a \leq b$ permits the simultaneous occurrence of a and b. Independence can be expressed with the requirement that if two prossets of a process differ only in that $a \leq b$ holds in one and $b \leq a$ in the other then the process also contains a prosset differing from those two in omitting both constraints, thereby expressing their independence. If however $a < b$ and $b < a$ hold in the respective prossets then this is understood to mean $ab + ba$, the mutually exclusive execution of a and b in either order.

A quite different way of drawing this distinction is with the notion of a ***higher dimensional automaton*** [54] or HDA based on combinatorial geometry. This approach abandons events and reverts to states. However the conception of state is an n-dimensional one in which a process consists of combinatorial cells whose dimension gives the number of ongoing events in that cell. While the 0-dimensional cells correspond to the ordinary notion of state, the 1-dimensional cells look more like transitions, while the higher-dimensional cells express concurrency and have no counterpart in ordinary automata theory. This approach amounts to an automata-theoretic formalization of Papadimitriou's geometric treatment of concurrency control [40], as well as ST-bisimulation [18] and deterministic asynchronous automata [71]. Higher dimensional automata have since been studied by many authors [10, 12, 19–23, 25, 26, 29, 63, 70, 72, 74], leading Eric Goubault to found a series of conferences on the geometry and topology of computation, GETCO, and a special issue of MSCS [24].

My own perspective on concurrency has evolved gradually over the past three decades [31, 47–61] by way of pomsets, prossets, event structures, higher dimensional automata, and finally Chu spaces. The last were so interesting in their own right as to distract me from process algebra in order to study their applications to mathematics for a few years before returning to their process algebra applications [63–65, 67]. [67] in particular made the observation that the usual two values 0 and 1 of Chu spaces, denoting event states of *ready* (not yet started) and *done*, could be extended with either or both of the intermediate value ⌐ of *transition* and the alternative value $\times$ of *cancelled*. These permit respectively higher-dimensional automata and what we now call cancellation automata to be represented as Chu spaces. In particular we show how van Glabbeek's example of a higher dimensional automaton not expressible as a Petri net [75, Fig.11] can be expressed instead as a pure cancellation automaton, one with no higher dimensional cells.

This paper serves the dual purposes of an up-to-date tutorial on the representation of processes as Chu spaces over $K = \{0, 1, ⌐, \times\}$, which we propose to call linear processes, and its relationship to these earlier models, along with an update on recent work. The Chu representation of concurrent processes is not as well known in concurrency circles as I feel it should be, despite having been in the literature for two decades [8, 9, 30, 31], justifying the tutorial part. The emphasis here is more on rationale, intuition, definitions, and perspective and less on a formal theorem-and-proof development of the theory; for more in-depth technical details see [62] for the relevant theory of Chu spaces and [67] for more on process algebra based on transition and cancellation.

Algebra traditionally starts with operations and laws forming a theory, e.g. the theory of commutative rings or of fields, leaving the values to emerge as the elements of models of the theory. Useful algebra however is often motivated by its intended or primary model, e.g. the ring of integers or the field of rationals, and for this reason it is preferable to begin with the intended values as motivation for the operations, and to let both drive the laws rather than vice versa. This is the approach we follow here in starting with Chu spaces as the values, then explicitly defining operations on them, and lastly considering what form the laws should take and what they are.

2 Chu Spaces as Generalized Linear Algebra

The previous section noted the diverse notions of process, some based on events, such as pomsets and event structures, others on states, such as information systems, synchronization trees, and higher-dimensional automata. A common core compatible with both kinds needs somehow to cater for and reconcile both events and states. To unify such diversity would appear to call for a complex framework; in particular one would not expect the simplest conceivable framework meeting the brief desiderata of the previous sentence to be up to the job on its own. Nevertheless that is what we propose and study here.

We take for our core notion of (unlabeled) process simply a set of events, a set of states, and a binary relation between them, and nothing else. We write these as respectively A, X, and $r : A \times X \to K$, where K is a set (such as $\{0, 1\}$) making r a K-valued binary relation. A map $h : P \to Q$ transforming process $P = (A, r, X)$ into process $Q = (B, s, Y)$ is defined as an ***adjoint*** pair $(\hat{h}, \check{h})$ of functions $\hat{h} : A \to B$, $\check{h} : Y \to X$, meaning one that satisfies $s(\hat{h}(a), y) = r(a, \check{h}(y))$ for all $a \in A$ and $y \in Y$. The maps from P to Q are all and only those pairs of functions satisfying these conditions.

In general such a structure is called a Chu space over K, with the category of such and their maps being denoted by $\mathbf{Chu}(\mathbf{Set}, K)$. Chu spaces can be specialized to particular applications by a suitable choice of K. Before developing the process concept further, by way of background we first consider other areas whose objects can be organized as Chu spaces.

The paradigmatic example is linear algebra over a given field k. This is more than just an analogy: as pointed out by Y. Lafont [34, 35] it is the special case $K = |k|$, the set of elements of the field. Each vector space V is represented as the Chu space $\tilde{V} = (|V|, r, |V^*|)$ whose points are the vectors of V, whose states are its functionals or dual points, namely the linear transformations comprising the vectors of the vector space $V^* = k^V$ (treating k as a one-dimensional vector space over k) and $r : V \times V^* \to K$ is the inner product for V, satisfying $r(v, g) = g(v)$ for each vector $v \in V$ and functional $g \in V^*$. In mathematics inner product $r(v, g)$ is customarily written (v, g), in physics as $\langle g|v\rangle$ (bra $\langle g|$ and ket $|v\rangle$). It can be shown that each linear transformation $h : U \to V$ is represented uniquely as the pair $(h, \lambda g.gh) : \tilde{U} \to \tilde{V}$ constituting a morphism of the Chu spaces representing respectively U and V. That is, the category $\mathbf{Vct}_k$ of vector spaces over k and their linear transformations fully embeds in the category $\mathbf{Chu}(\mathbf{Set}, K)$, creating a bijection between homsets $\mathbf{Vct}_k(U, V)$ and $\mathbf{Chu}(\mathbf{Set}, K)(\tilde{U}, \tilde{V})$.

As another instance, Y. Lafont [34, 35] has further pointed out that the points and open sets of a topological space S can be treated by analogy with respectively the vectors and functionals of a vector space, with r taken to be the two-valued relation of membership of a point in an open set. The counterpart of the one-dimensional space is the Sierpinski space with two points and three open sets, while each continuous function $h : S \to T$ is represented uniquely as the pair $(h, h^{-1}) : \tilde{S} \to \tilde{T}$ where h^{-1} is the inverse image function associated to h. We give a great many more such examples elsewhere [62, 66].

The term "linear" is also motivated by Girard's linear logic, LL, a substructural logic applicable to sequent-based proof theory. The "multiplicatives" of LL, namely perp $P^\perp$, tensor $P \otimes Q$, its De Morgan dual $P \parr Q = (P^\perp \otimes Q^\perp)^\perp$ and the multiplicative units 1 and $\perp$, constituting multiplicative linear logic MLL, make essentially the same connection with Boolean algebra via the Curry-Howard isomorphism as do their counterparts for linear process algebra. Furthermore the extension of MLL to MALL with the "additives" $P \oplus Q$ and $P \& Q$ of LL, as respectively direct sum (coproduct) and direct product as notated by Girard (we will write $P \& Q$ as the more customary $P \times Q$), also find application in both LL and LPA.

3 Linear Processes

In this section we define processes as structures. Just as the Chu representation of linear algebra over a field k took $K = |k|$, and of topological spaces, $K = \{0, 1\}$, so do we define linear processes Chu spaces over the set $K = \{0, ɾ, 1, \times\}$. Organizing processes as Chu spaces makes events and states equally primary, paralleling Hamilton's reorganization of Newton-Langrange mechanics in 1837 by putting position and momentum on the same level.[2]

The elements of K constitute the four possible states an event can be in, namely *ready* 0 (i.e. not yet started), *transition* ɾ, *done* 1, and *cancelled* $\times$.[3] The intuitive meaning of transition is as in "Shh, the event is in progress", while that of cancellation is as in "Sorry but the event has been cancelled."

Thinking of these four as *local* states, we interpret the value of $r(a, x)$ as the local state of event a in state x. The latter is a *global* state or state vector in the sense that it can be interpreted extensionally via the function $X \to (A \to K)$ mapping each x in X to the function $\lambda a.r(a, x) : A \to K$, which we call the ***extension*** of x. Dually the extension of each event a is the function $\lambda x.r(a, x) : X \to K$. We can refer to a row of a matrix either intensionally by its index a or extensionally by the row itself, and dually for columns.

Following Barr's terminology [4] we call a Chu space ***extensional*** when if two columns have the same extension then they have the same intension; that is, there are no repeated extensional columns. Topological spaces can be understood as extensional Chu spaces by identifying their open sets with the extensions of states. Dually a Chu space is ***separated*** when two rows with the same extension

[2] Hamiltonian mechanics took 90 years to catch on in physics; hopefully event-state symmetry will not take that long!

[3] This is sufficient for a theory of ideal processes. In practice processes need to be abortable, which a fifth state, *aborted*, could address. We leave this to future work.

have the same intension; this corresponds to the notion of a T_0 topological space. A ***biextensional*** Chu space is one that is both extensional and separated, all rows and columns distinct. The above representation of vector spaces as Chu spaces is biextensional.

Viewed as event structures, Chu spaces are unlabeled. As with the theory of event structures we draw the distinction between events and actions. An event constitutes an instance of an action; it can happen only once, whereas an action can happen many times.

We take events to be more basic than actions on the ground that the order in which things happen in a process is an ordering of events, not of actions. This is not to say that actions are unimportant but that the structure of events independently of their labels is already of considerable interest in its own right.

A labeled process over an alphabet Λ of actions is a pair (P, λ) where $P = (A, r, X)$ is an unlabeled process and $\lambda : A \to \Lambda$ labels each event a with the action $\lambda(a)$ of which a is an instance.

4 Processes as Transformable Entities

The preceding section defined a process over a set K as a structure (A, r, X) where $r : A \times X \to K$. In this section we instead define processes analogously to how Zermelo-Fraenkel set theory, ZF, defines sets. Whereas all individuals of a model of ZF are sets, those of our theory are of two sorts, processes $P, Q, \ldots$ forming a class $\mathcal{P}$ and process transformations or maps $h : P \to Q$ forming a class $\mathcal{M}$. And whereas the language of ZF consists of a single binary relation $\in$ of membership on a homogeneous domain of sets, ours consists of one binary operation, three unary operations, and two constants. Besides permitting a more axiomatic definition, this perspective distinguishes the events of a process P from its states by exhibiting them as maps respectively to and from P[4] and makes the event-state schizophrenia of the elements of K more explicit by exhibiting them as states of the generic one-event process $\mathbf{1}$, which of course they are, as well as events of K, which of course they are.

The binary and unary operations are just those for a category, namely composition $m : \mathcal{M}^2 \to M$, source and target $s, t : \mathcal{M} \to \mathcal{P}$ and identity $i : \mathcal{P} \to \mathcal{M}$. Moreover they satisfy the usual laws, namely $s(i(P)) = t(i(P)) = P$, $m(k, h)$ or kh for short is defined just when $s(k) = t(h)$, $s(kh) = s(h)$, $t(kh) = t(k)$, $(kh)j = k(hj)$ (associativity), and $i(P)$ or 1_P for short is both a left and right identity for composition.

As usual a map $h : P \to Q$ is an isomorphism when it has an inverse $h^{-1} : Q \to P$ in the sense that both $h^{-1}h$ and hh^{-1} are identity maps, namely 1_P and 1_Q respectively. Processes are isomorphic when there is an isomorphism between them.

[4] Early on [30, 31] we took events and states to be respectively columns and rows for consistency with linear algebra, which traditionally identifies the points and functionals of a vector space with respectively columns (maps to the space) and rows (maps from it), but found this unnatural in thinking about processes and subsequently reversed the correspondence.

The only process-specific part of the language consists of two constant processes $\mathbf{1}$ and K, satisfying four axioms, A1-A4, the first two of which are elementary (first order).

Definition 1. *A process P is* ***rigid*** *when the only map $P \to P$ is the identity 1_P.*

Axiom A1. *The processes $\mathbf{1}$ and K are rigid.*

Definition 2. *An* ***event*** *is a map from $\mathbf{1}$, while a* ***state*** *is a map to K.*

Events to P and states from P are considered to belong to P, as in "event (state) of P." Variables ranging over events and states are written $a, b, \ldots$ and $x, y, \ldots$ respectively. An ***ordinary*** map is one that is neither an event nor a state.

For convenience we denote by A_P and X_P the sets of respectively events and states of P. The following two propositions are routine.

Proposition 1. *$\mathbf{1}$ has one event, while K has one state.*

Proposition 2. *The events of K are precisely the states of $\mathbf{1}$.* (So they have a dual identity as events and states.)

Definition 3. *The state of an event a of P in state x of P is the state xa of $\mathbf{1}$* (and an event of K).

We distinguish states of processes from states of events by considering them respectively global and local states. By proposition 2 local states are events of K. Composition of states with events of a process is the axiomatic counterpart of r in the structural definition (A, r, X).

Given any map $h : P \to Q$ we define its ***left action*** $\hat{h}$ to be the function $\lambda a.ha$ mapping each event a of P to the event ha of Q, and its ***right action*** $\check{h}$ to be the function $\lambda y.yh$ mapping each state y of Q to the state yh of P. The types of these actions are respectively $\hat{h} : A_P \to A_Q$ and $\check{h} : X_Q \to X_P$.

Proposition 3. *For any map $h : P \to Q$, its left and right actions satisfy $x\hat{h}(a) = \check{h}(x)a$ for all $a \in A_P$ and $x \in X_Q$.*

Functions $A_P \to A_Q$ and $X_Q \to X_P$ satisfying this condition are said to be an ***adjoint pair*** of functions between P and Q.

Proof. The two sides expand to respectively $x(ha)$ and $(xh)a$, which are equal by associativity.

Definition 4. *Two maps $h, k : P \to Q$ with the same left and right actions are called* ***equivalent****.*

Axiom A2. (Extensionality) *Equivalent maps are equal.*

Thus far all definitions and propositions have been equivalent to elementary or first order ones; the second order constructs are inessential and can be translated into first order ones. By introducing two process operations $P \otimes Q$ and $P \multimap Q$ we could continue the ZF analogy with further elementary axioms. However a hybrid approach involving counterfactuals (whose explication in the absence of a self-contained foundation requires falling back on ZF) allows this transformation-based axiomatization to be completed much faster using just two more axioms, neither one elementary in the sense of first order logic but both elementary in the sense of being intuitively clear. We leave to another occasion the development of a purely first order theory of processes replacing A3 and A4.

The purpose of Axiom A3 is to ensure that all possible maps are present between the processes of the model. We state it as a counterfactual which contemplates the possibility of additional ordinary maps not already in whatever the model happens to be, and denies this possibility in a positive way. It is counterfactual in that it refers to a map that does not yet exist, a notion absent from first order logic. "Ordinary" is essential here since adding new events or states to a process would make it a different process.

Axiom A3. (No new maps) *Any new ordinary map is equivalent to an old one.*

A3 together with A2 implies that $\mathcal{M}$ has no proper extension. That is, $\mathcal{M}$ is maximal and the contemplated counterfactual is impossible.

The following establishes uniqueness up to isomorphism of such a maximal $\mathcal{M}$, namely the set of all pairs of actions.

Proposition 4. (Density) *For any two processes P, Q for which P is not* $\mathbf{1}$ *and Q is not K, every adjoint pair of functions between P and Q is the pair of actions of some map $g : P \to Q$.*

This notion of density is due to Gabriel and Ulmer [13] by analogy with density of the rationals in any extension thereof, namely any Archimedean field. Here it means, informally speaking, that all possible maps between two processes are present.

Proof. Suppose some adjoint pair between P and Q is realized by no map. Then a map $g : P \to Q$ with this pair as its actions can be adjoined, along with all required composites $hgf : P' \to Q'$ not already present where $f : P' \to P$ and $h : Q \to Q'$. No other maps than these need be adjoined (there is no chain reaction) since by associativity any map of the form $h'(hgf)f'$ is of the form $(h'h)g(ff')$ and hence is already adjoined. All compositions are uniquely determined by the actions of g and those of the f's and h's.

Since the possible pairs of actions between two processes form a set this maximum is well-defined.

Axiom A4 similarly depends on the counterfactual notion of extending $\mathcal{P}$ (whatever processes exist in the model at hand) with additional processes, subject to maintaining the preceding axioms. When a process is added, its events and states and how they compose as maps of A_K are considered part of the addition, and these are thereafter left undisturbed by any further extensions of either $\mathcal{M}$ or $\mathcal{P}$.

As part of this addition, all new morphisms needed to satisfy A3 are added. This is a weak denial of the impossibility of new processes in that it takes "new" to mean new up to isomorphism.

Axiom A4. (No new processes) *Any new process is isomorphic to an old one.*

As defined earlier P and Q are isomorphic when there exists an isomorphism $h : P \to Q$. We need A3 for this, without which processes that an external observer would judge isomorphic might nevertheless lack the requisite isomorphism witnessing their isomorphism.

In the following "equivalent" means in the sense of equivalent categories.

Proposition 5. (Completeness) *All models of these axioms are equivalent.*

Proof. For any two sets A, X and any function $r : A \times X \to A_K$ there exists a process P and bijections $\alpha : A \to A_P$, and $\omega : X \to X_P$, such that $r(a, x) = \omega(x)\alpha(a)$ (composition). It follows that every model of these axioms is equivalent to the category of Chu spaces over A_K, as defined in the preceding section, when its morphisms from (A, r, X) to (B, s, Y) are taken to be all adjoint pairs (f, g) of maps $f : A \to B$ and $g : Y \to X$, that is, maps satisfying $s(f(a), y) = r(a, g(y))$ for all $a \in A$ and $y \in Y$.

5 Linear Process Algebra

We turn now from linear process semantics, what processes are, to process algebra, how to name them compositionally and reason about them.

The operations of Linear Process Algebra, LPA, are orthocurrence $P \otimes Q$, concurrence $P||Q$, sequence $P;Q$ or just PQ, and choice $P + Q$. In defining operations we assume that all processes are extensional (states with equal extensions are equal, i.e. no repeated columns). The extensional collapse of a process P is the result of making P extensional by identifying states of P with equal extensions. Since some of the operations below do not necessarily preserve extensionality we enforce it by automatically collapsing extensionally the result of every operation.

Dual $P^\perp$. Duality makes the connection between processes viewed as a schedule of events and as an automaton comprised of states. Given $P = (A, r, X)$, $(A, r, X)^\perp$ is defined as (X, r', A) where $r'(x, a) = r(a, x)$, that is, transpose. Transpose also applies to process maps, with the transpose of $h : P \to Q$ being $h^\perp : Q^\perp \to P^\perp$, where h and $h^\perp$ have the same actions in the sense of Section 4 but with left and right simply interchanged.

Duality has no operational interpretation, but rather serves to convert an event-oriented process, meaning one whose events and states transform respectively covariantly and contravariantly, into a state-oriented process for which events and states transform respectively contravariantly and covariantly.

Orthocurrence $P \otimes Q$. This is the fundamental interaction operator of linear process algebra. Although it appeared in our work almost as early as did pomsets [11, 31, 51, 53], it is at the heart of both Barr's category-theoretic notion of a

$*$-autonomous category [3], and Girard's proof-theoretic notion of linear logic developed independently several years after Barr. [16].

Given two processes $P = (A, r, X)$ and $Q = (B, s, Y)$, their ***orthocurrence*** $P \otimes Q$ is the process $(A \times B, t, Z)$. Here Z is the set of all functions $z : A \times B \to K$ such that
(i) for each $b \in B$, $\lambda a.z(a, b)$ is a state of P, and
(ii) for each $a \in A$, $\lambda b.z(a, b)$ is a state of Q,
while $t : (A \times B) \times Z \to K$ is defined as $t((a, b), z) = z(a, b)$. Each z may be thought of as an $A \times B$ crossword whose rows are states of Q (the "across" dictionary) and whose columns are the states of P (the "down" dictionary).

Besides interaction, othocurrence $P \otimes Q$ can also be understood as dual to the *observation* $P \multimap Q^{\perp}$ of states of process Q (events of $Q^{\perp}$) from vantage points of process P, or by symmetry the observation $Q \multimap P^{\perp}$, giving a sense in which observation is as symmetric as orthocurrence [64].

Concurrence $P||Q$. This is the independent or noninteracting parallel behavior of P and Q. Given two processes $P = (A, r, X)$ and $Q = (B, s, Y)$, their ***concurrence*** $P||Q$ is the process $(A + B, t, X \times Y)$. Here $A + B$ denotes the disjoint or marked union of A and B, where the marking indicates for each event of $A + B$ whether it came from A or B (e.g. by defining $A + B = A \times \{1\} \cup B \times \{2\}$) while $t(a, (x, y)) = r(a, x)$ and $t(b, (x, y)) = s(b, y)$ where a and b denote events of $A + B$ coming from respectively A and B.

Initial states. The initial local or event state is 0 or *ready.* The initial global state is the all-zero state vector: all events are in their ready state 0.

Final states. The two final local states are 1 and $\times$. A state is final just when all its events are in a final event state.

This definition is facilitated by the cancel state. Without it the event structure literature has struggled with the concept of final state, termination, and sequence. One might suppose that a final state could be defined simply as one with no successor state. This however fails to represent a process that may choose nondeterministically to halt or continue. For example the process $a + \emptyset$ that chooses to do either a or nothing has a state in which a is ready, but that state cannot be final because it has 1 as a successor state. While solutions have been proposed, none are as simple as merely allowing a to enter the cancelled state to indicate termination.

Cancelled states. A state is cancelled when all its events are cancelled. The typical application is to the definition of choice $P + Q$, whose first step is to cancel one of P or Q simultaneously with starting the other.

Sequence PQ. Given two processes $P = (A, r, X)$ and $Q = (B, s, Y)$, their ***sequence*** PQ is defined as for $P||Q$ but with only those states (x, y) of $X \times Y$ for which either y is initial or x is final. When y is initial we consider P to be happening when in state (x, y), while when x is final we consider Q to be happening.

If Q has no initial state (usually not the case in practice) then P cannot run and is understood to be in one of its final states. If P has no final state then Q

cannot run. If P has n final states x_i then PQ has n copies (x_i, y) of each state y of Q.

Choice $P + Q$. Given two processes $P = (A, r, X)$ and $Q = (B, s, Y)$ with both A and B nonempty, their ***choice*** $P + Q$ is the process $(A + B, t, \{*\} + X' + Y')$ where X' and Y' are X and Y less their respective initial states, $t(a, *) = t(b, *) = 0$ (making $*$ the initial state of $P + Q$), $t(a, x) = r(a, x)$, $t(b, y) = s(b, y)$, and $t(a, y)$ and $t(b, x)$ are the cancelled state ($\times$ if available, otherwise 0). Operationally, $P + Q$ begins in the initial state (all events of both P and Q ready) and then simultaneously cancels all the events of one of P or Q and begins the other.

Constants $\emptyset$ *and* $\mathbf{1}$. The process $\emptyset$ is $(0, !, 1)$, consisting of no events and one state. It is the unit for both concurrence and sequence, that is, $a||\emptyset = a\emptyset = \emptyset a = a$. However $\emptyset$ is not the unit for choice because $P + \emptyset$ creates a state in which all events of P are cancelled.

The process $\mathbf{1}$ is as defined in Section 4. As a Chu space it can be taken to be $(\{*\}, s, K)$ where $s(*, k) = k$. Up to isomorphism $\mathbf{1}$ is the unit for orthocurrence: when (A, r, X) "flows through" $\mathbf{1}$, $A \times \{*\}$ is isomorphic to A and the states of $A \otimes \mathbf{1}$ are in bijection with those of A.

6 The Curry-Howard Correspondence with Boolean Algebra

The core of linear process algebra is orthocurrence $P \otimes Q$ as interacting concurrency, which we distinguish from concurrence $P||Q$ as noninteracting concurrency, parallel play as kindergarten teachers call it. Orthocurrence together with the involution $P^\perp$ share essential features with Boolean conjunction $x \wedge y$ and complement $\neg y$.

The connection is made via the Curry-Howard correspondence[5] between logical values and mathematical objects. In this case the logical values may be taken to be 0 and 1 while the objects are linear processes. Conjunction and negation of the former correspond respectively to orthocurrence and dual of the latter.

Many of the logical laws involving terms built with these two logical operations have their counterpart as natural isomorphisms between functors built from these two process operations. In particular associativity and commutativity of conjunction carry over, the latter as a symmetry $P \otimes Q \cong Q \otimes P$, but idempotence has no counterpart. Double negation $\neg\neg x = x$ does carry over, with $P^{\perp\perp}$ being not only isomorphic but equal to P. And just as $x \vee y$ is definable by De Morgan's law as $\neg(\neg x \wedge \neg y)$, so is Girard's *par* operation $P \parr Q$ definable as $(P^\perp \otimes Q^\perp)^\perp$. The meaning of $P^\perp$ in LPA is P viewed as an automaton consisting of (covariantly transforming) states instead of as a schedule consisting of events, while the meaning of $P \parr Q$ is just the automaton counterpart of orthocurrence.

[5] This is commonly called the Curry-Howard isomorphism but since it is not technically an isomorphism we prefer to call it a correspondence.

7 Example Terms

We now consider the behavior of the operations on atomic processes, showing how they compose in certain cases to produce larger terms. Table 1 lists a dozen terms and the linear processes they denote.

Table 1. Example LPA expressions

a	a	0⌐1	$a+\emptyset$	a	0⌐1×
ab	a	0⌐111	$a+b$	a	0⌐1××
	b	000⌐1		b	0××⌐1
$a\|\|b$	a	0⌐10⌐10⌐1	$ab+ba$	a	0⌐1010⌐1
	b	000⌐⌐⌐111		b	000⌐⌐111
$a(b+c)$	a	0⌐11111	$ab+ac$	a	0⌐111⌐111
	b	000⌐1××		b	000⌐1××××
	c	000××⌐1		c	0××××00⌐1
$(b+c)a$	a	000⌐100⌐1	$ba+ca$	a	000⌐100⌐1
	b	0⌐111××××		b	0⌐111××××
	c	0××××⌐111		c	0××××⌐111
$ab\otimes cd$	ac	0⌐11111111111	$(a+b)\otimes(c+d)$	ac	0⌐⌐11××××
	ad	00000⌐⌐⌐11111		ad	0××××⌐⌐11
	bc	000⌐10⌐10⌐111		bc	0××××⌐1⌐1
	bd	00000000000⌐1		bd	0⌐1⌐1××××

Cancellation distinguishes $a+\emptyset$ from a by adjoining a fourth state to a allowing it to be cancelled. Hence a has only one final state while $a+\emptyset$ has two.

Sequence ab and choice $a+b$ each have five states. These are performed sequentially for ab, while in $a+b$ they form two branches each with three states, the initial state of which is common to both branches.

Processes $a||b$ and $ab+ba$ are almost identical, the one difference being that ⌐⌐ is a state of the former but not of the latter. Thus $a||b$ has a two-dimensional state while $ab+ba$ does not, but otherwise has the same states of dimension 0 and 1 as $a||b$.

Process $a(b+c)$ does not cancel either b or c until after a is done. At that point the state is 100, which can be viewed as the initial state of $b+c$. One of b or c is then cancelled while simultaneously the other gets under way in the transition state, and then is done, for a total of 7 states. Process $ab+ac$ on the other hand cancels one of b or c as soon as a enters its transition state, after which it behaves like an ordinary sequence. There are thus two branches with 5 states each, but with the initial state shared so that there are only 9 rather than 10 states. Hence $a(b+c)$ and $ab+ac$ are distinct.

Processes $(b+c)a$ and $ba+ca$ however are the same: both begin by cancelling one of b or c while beginning the other. This gives them both two branches, branching at the root with each branch having 5 states as for $ab+ac$.

The orthocurrence $ab \otimes cd$ involves no cancellation. However there is one instance of concurrency, namely ad with bc, where ac is done and bd is ready.

One example of this situation is a train schedule involving two sequential trains a then b passing through two stations c then d. The pair ac is the event of train a arriving at station c. When ac is in transition the train is standing at the station; passing to done corresponds to the train having left the station. The only opportunity for concurrency here is when the first train is standing at the second station while the second train is at the first station.

Another example of $ab \otimes cd$ is Allen's 13 configurations of a pair of intervals sliding past each other [1], with a and b denoting the endpoints of one interval and c and d those of the other. Allen's 13 configurations correspond in the evident way to the 13 states shown in Table 1. In particular aligning the two intervals at both ends corresponds to the one two-dimensional state 1┏┏0. Rodriguez and Anger [69] have studied extensions of Allen's configurations to handle richer notions of time such as relativistic time, accomplished by suitably enlarging the local event set K. With one extension they characterize as branching time the 13 states extend to 29, with their relativistic one it extends to 82 states, see [67] for further details.

8 Laws

As indicated in Section 6, the Curry-Howard counterpart of many (but not all) of the equations of Boolean algebra are natural isomorphisms between terms involving orthocurrence (*tensor*), duality (*perp*), and dual orthocurrency (*par*). There are in addition laws governing concurrence, sequence, and choice, but already just those governing the first three mentioned operations raise interesting questions.

Perhaps the most important question is, what is the Curry-Howard counterpart of logical truth in the LPA setting? One might suppose that the same question would arise for linear logic and therefore serve as a guide. However Girard has taken the position that truth is merely that which proof establishes, as opposed to being definable independently. Since LPA deals with processes rather than proofs, this view would appear to make no sense for LPA.

Yet there is one point of commonality: the classical notion of truth does arguably not make sense for LPA. The usual conception of truth for a proposition, at least for Boolean logic, entails a binary decision. Such a decision is necessarily centralized, with data from sensors being collected at one point to arrive at a binary determination.

The Curry-Howard counterpart of a proposition is a process. Processes need not be local as they can be distributed over an arbitrarily large area or volume. This is not consistent with logical decision-making as a centralized notion as decisions must be made locally if they are to be timely. This in turn implies that decision itself should be a concurrent notion.

We therefore propose to dispense with the traditional notion of propositions as special entities that are true or false and simply define a proposition to be a process. We take reasoning to be an extension of behavior that introduces events of a propositional or judgmental nature above and beyond the ordinary events of behavior. There can be many of these running concurrently in a distributed fashion, with no requirement of global coordination at any point.

The benefit of this approach is that reasoning can be absorbed into the framework without making special provision for it as a distinct notion. Reasoning becomes simply a kind of parallel behavior.

We take reasoning to be the transformation of one process into another, in its most general sense. Transformation acts on terms, and terms are realized by functors. When all functors are covariant a transformation can be defined simply as a natural transformation. In the presence of contravariant functors, logic of this kind is more delicate. Dinatural transformations have been proposed for this [36]; however Chapter 6 of [62] points out difficulties with dinaturality that are overcome using binary logical transformations [43].

When the processes being so transformed represent behavioral rather than propositional information, transformations can be regarded as serving simply to establish program equivalence. Processes incorporating distributed propositional information may convey more nuanced decision-oriented information, much in the manner of couriers carrying information between locations and multiple local headquarters planning for their immediate neighborhood. In short, very much how reasoning is carried out in the real world, namely by many individuals, organizations, and machines, in a distributed fashion.

This conception of distributed reasoning is not at all well worked out here, and we hope to sharpen these ideas more satisfactorily in due course.

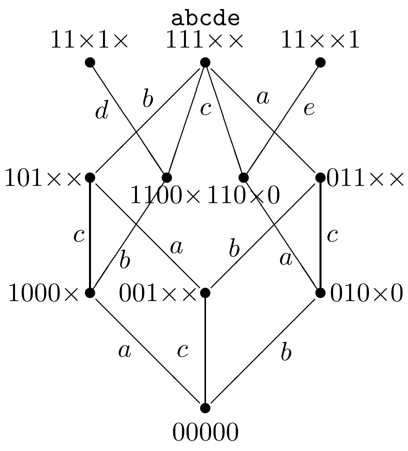
9 Beyond Petri Nets

In [75, Fig.11] van Glabbeek gives an example of a higher-dimensional automaton expressing a process that is not expressible as a Petri net. The following story relocates van Glabbeek's process to a more rural setting than the race across the conference podium that enlivened Rob's presentation of this example at EXPRESS'04.

Alphonse and Gaston are walking abreast along a path when they come to a gate that seems stuck half-open, obliging them to pass through it in single file. Neither one wishing to be the first to emerge, they try to push the gate open as they pass through in an attempt to emerge together. The three possible outcomes, c, d, and e, are that they succeed (c), or that they fail and one of Alphonse (d) or Gaston (e) emerges first. That is, exactly one of the events c, d, and e must occur. Taking a and b to be the events of respectively Alphonse and Gaston entering the opening (which could happen either before, after, or instead of succeeding in opening the gate), represent this scenario suitably.

Van Glabbeek proposed the following 11-state automaton, which however he interpreted as a higher-dimensional automaton with 31 states when the 15 one-dimensional and 5 two-dimensional states are counted. Our depiction here labels the 11 states to exhibit it as a pure cancellation automaton, one with no higher-dimensional cells. We give it in both forms, matrix and visual (its Hasse diagram).

$$\begin{array}{c|ccccccccccc|}
\hline
a & 0&1&0&0&1&1&1&0&1&1&1\\
b & 0&0&0&1&0&1&1&1&1&1&1\\
c & 0&0&1&0&1&0&0&1&\times&1&\times\\
d & 0&0&\times&\times&\times&0&\times&\times&1&\times&\times\\
e & 0&\times&\times&0&\times&\times&0&\times&\times&\times&1\\
\hline
\end{array}$$



The success of this representation depends on the observation that as soon as either party has entered the opening at least one of d or e may be immediately cancelled. Without cancellation, the state of both a and b being in the opening with the gate still stuck would be 110 regardless of their single-file order. With cancellation, exactly one of d or e is cancelled in that situation, refining 110 into the two states 110×0 and 1100×, thereby creating the natural five-state automaton for $ab+ba$ which is what the stuck gate obliges, thereby splitting the "ground floor" $c = 0$ of what would otherwise be the abc cube. The top floor, $c = 1$, gate open, represents $a||b$.

The five two-dimensional cells that naturally suggest themselves, namely two instances of $a||c$ and $b||c$ and one of $a||b$ (so five out of the six faces of the abc cube), could certainly be added, making this a full-blown higher-dimensional cancellation automaton with $11 + 15 + 5 = 31$ cells. However when $a||b$ is understood by default to include all possible higher-dimensional cells, that is, no mutual exclusion, it is not necessary.

Van Glabbeek's example raises an important distinction that we have glossed over so far. Table 1 makes the distinction between $a||b$ and $ab+ba$ one of mutual exclusion: the latter forbids the state ⌐⌐. The role of events d and e here is to record the primacy of respectively a and b in the choice implied by $ab + ba$. In the absence of c the above automaton simplifies to that on the left below. Alternatively we can be more faithful to our disjoint-union definition of $P + Q$

than thus far and obtain an isomorphic automaton in terms of marked copies of just a and b as on the right.

$$\begin{array}{c|ccccc|} a & 0 & 1 & 1 & 0 & 1 \\ b & 0 & 0 & 1 & 1 & 1 \\ d & 0 & 1 & 1 & \times & \times \\ e & 0 & \times & \times & 1 & 1 \end{array} \qquad \begin{array}{c|ccccc|} a & 0 & 1 & 1 & \times & \times \\ b & 0 & 0 & 1 & \times & \times \\ a' & 0 & \times & \times & 1 & 1 \\ b' & 0 & \times & \times & 0 & 1 \end{array}$$

References

1. Allen, J.F.: Towards a general theory of action and time. Artificial Intelligence 23, 123–154 (1984)
2. Baeten, J.C.M., Weijland, W.P.: Process Algebra. Cambridge University Press, Cambridge (1990)
3. Barr, M.: ∗-Autonomous categories. Lecture Notes in Mathematics, vol. 752. Springer, Heidelberg (1979)
4. Barr, M.: ∗-Autonomous categories and linear logic. Math Structures in Comp. Sci. 1(2), 159–178 (1991)
5. Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. Information and Control 60, 109–137 (1984)
6. Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.): Handbook of Process Algebra. Elsevier (North-Holland), Amsterdam (2000)
7. Brookes, S.D., Hoare, C.A.R., Roscoe, A.D.: A theory of communicating sequential processes. Journal of the ACM 31(3), 560–599 (1984)
8. Brown, C., Gurr, D.: A categorical linear framework for Petri nets. In: Mitchell, J. (ed.) Logic in Computer Science, pp. 208–218. IEEE Computer Society, Los Alamitos (June 1990)
9. Brown, C., Gurr, D., de Paiva, V.: A linear specification language for Petri nets. Technical Report DAIMI PB-363, Computer Science Department, Aarhus University (October 1991)
10. Buckland, R., Johnson, M.: Echidna: A system for manipulating explicit choice higher dimensional automata. In: Nivat, M., Wirsing, M. (eds.) AMAST 1996. LNCS, vol. 1101, Springer, Heidelberg (1996)
11. Casley, R.T., Crew, R.F., Meseguer, J., Pratt, V.R.: Temporal structures. Math. Structures in Comp. Sci. 1(2), 179–213 (1991)
12. Fajstrup, L., Goubault, E., Raussen, M.: Detecting deadlocks in concurrent systems. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 332–347. Springer, Heidelberg (1998)
13. Gabriel, P., Ulmer, F.: Lokal präsentierbare Kategorien. Lecture Notes in Mathematics, vol. 221. Springer, Heidelberg (1971)
14. Gaifman, H., Pratt, V.R.: Partial order models of concurrency and the computation of functions. In: Proc. 2nd Annual IEEE Symp. on Logic in Computer Science, Ithaca, NY, pp. 72–85 (June 1987)
15. Ginsburg, S., Spanier, E.H.: Mappings of languages by two-tape devices. Journal of the ACM 12, 423–434 (1965)
16. Girard, J.-Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
17. Gischer, J.L.: The equational theory of pomsets. Theoretical Computer Science 61, 199–224 (1988)

18. van Glabbeek, R.J., Vaandrager, F.W.: Petri net models for algebraic theories of concurrency. In: de Bakker, J.W., Nijman, A.J., Treleaven, P.C. (eds.) PARLE 1987. LNCS, vol. 259, pp. 224–242. Springer, Heidelberg (1987)
19. Goubault, E.: Homology of higher-dimensional automata. In: CONCUR 1993. LNCS, vol. 630, pp. 254–268. Springer, Heidelberg (1993)
20. Goubault, E.: The Geometry of Concurrency. PhD thesis, École Normale Supérieure (1995)
21. Goubault, E.: Schedulers as abstract interpretations of hda. In: Proc. of PEPM 1995, La Jolla, ACM Press, New York (June 1995)
22. Goubault, E.: Durations for truly-concurrent actions. In: Riis Nielson, H. (ed.) ESOP 1996. LNCS, vol. 1058, pp. 173–187. Springer, Heidelberg (1996)
23. Goubault, E.: A semantic view on distributed computability and complexity. In: Proceedings of the 3rd Theory and Formal Methods Section Workshop, Imperial College Press, London (1996)
24. Goubault, E.: Geometry and concurrency. Mathematical Structures in Computer Science, Special Issue 10(4), 409–573 (7 papers) (2000)
25. Goubault, E., Cridlig, R.: Semantics and analysis of Linda-based languages. In: Cousot, P., Filé, G., Falaschi, M., Rauzy, A. (eds.) WSA 1993. LNCS, vol. 724, pp. 72–86. Springer, Heidelberg (1993)
26. Goubault, E., Jensen, T.P.: Homology of higher dimensional automata. In: Cleaveland, W.R. (ed.) CONCUR 1992. LNCS, vol. 630, pp. 254–268. Springer, Heidelberg (1992)
27. Grabowski, J.: On partial languages. Fundamenta Informaticae IV(2), 427–498 (1981)
28. Greif, I.: Semantics of Communicating Parallel Processes. PhD thesis, Project MAC report TR-154, MIT (1975)
29. Gunawardena, J.: Homotopy and concurrency. EATCS Bulletin 54, 184–193 (1994)
30. Gupta, V.: Chu Spaces: A Model of Concurrency. PhD thesis, Stanford University, Tech. Report (September 1994), `http://boole.stanford.edu/pub/gupthes.pdf`
31. Gupta, V., Pratt, V.R.: Gates accept concurrent behavior. In: Proc. 34th Ann. IEEE Symp. on Foundations of Comp. Sci., pp. 62–71 (November 1993)
32. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Boston (2000)
33. Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM 21(8), 666–672 (1978)
34. Lafont, Y.: The linear abstract machine. TCS 59, 157–180 (1988)
35. Lafont, Y., Streicher, T.: Games semantics for linear logic. In: Proc. 6th Annual IEEE Symp. on Logic in Computer Science, Amsterdam, pp. 43–49 (July 1991)
36. Lambek, J., Scott, P.: Introduction to Higher-Order Categorical Logic. Cambridge University Press, Cambridge (1986)
37. Mazurkiewicz, A.: Concurrent program schemes and their interpretations. Technical Report DAIMI Report PB-78, Aarhus University, Aarhus (1977)
38. Milner, R.: A Calculus of Communication Systems. LNCS, vol. 92. Springer, Heidelberg (1980)
39. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures, and domains, part I. Theoretical Computer Science 13, 85–108 (1981)
40. Papadimitriou, C.: The Theory of Database Concurrency Control. Computer Science Press, Rockville (1986)
41. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
42. Petri, C.A.: Fundamentals of a theory of asynchronous information flow. In: Proc. IFIP Congress 62, Munich, pp. 386–390. North-Holland, Amsterdam (1962)

43. Plotkin, G.D.: Lambda definability in the full type hierarchy. In: To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, pp. 363–373. Academic Press, London (1980)
44. Plotkin, G.D.: A structural approach to operational semantics. Technical Report Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark (1981); Reprinted with corrections in J. Log. Algebr. Program. (60-61), 17–139 (2004)
45. Pnueli, A.: The temporal logic of programs. In: 18th IEEE Symposium on Foundations of Computer Science, pp. 46–57 (October 1977)
46. Pratt, V.R.: Semantical considerations on Floyd-Hoare logic. In: Proc. 17th Ann. IEEE Symp. on Foundations of Comp. Sci., pp. 109–121 (October 1976)
47. Pratt, V.R.: Process logic. In: Proc. 6th Ann. ACM Symposium on Principles of Programming Languages, San Antonio, pp. 93–100 (January 1979)
48. Pratt, V.R.: On the composition of processes. In: Proceedings of the Ninth Annual ACM Symposium on Principles of Programming Languages (January 1982)
49. Pratt, V.R.: Position statement. Circulated at the Panel on Mathematics of Parallel Processes, chair A.R.G. Milner, IFIP-83 (September 1983)
50. Pratt, V.R.: The pomset model of parallel processes: Unifying the temporal and the spatial. In: Seminar on Concurrency. LNCS, vol. 197, pp. 180–196. Springer, Heidelberg (1984)
51. Pratt, V.R.: Some constructions for order-theoretic models of concurrency. In: Logics of Programs. LNCS, vol. 193, pp. 269–283. Springer, Heidelberg (1985)
52. Pratt, V.R.: Two-way channel with disconnect. In: The Analysis of Concurrent Systems: Proceedings of a Tutorial and Workshop. LNCS, vol. 207, pp. 110–111. Springer, Heidelberg (1985)
53. Pratt, V.R.: Modeling concurrency with partial orders. Int. J. of Parallel Programming 15(1), 33–71 (1986)
54. Pratt, V.R.: Modeling concurrency with geometry. In: Proc. 18th Ann. ACM Symposium on Principles of Programming Languages, pp. 311–322 (January 1991)
55. Pratt, V.R.: Arithmetic + logic + geometry=concurrency. In: Simon, I. (ed.) LATIN 1992. LNCS, vol. 583, pp. 430–447. Springer, Heidelberg (1992)
56. Pratt, V.R.: The duality of time and information. In: Cleaveland, W.R. (ed.) CONCUR 1992. LNCS, vol. 630, pp. 237–253. Springer, Heidelberg (1992)
57. Pratt, V.R.: Event spaces and their linear logic. In: Algebraic Methodology and Software Technology, Workshops in Computing, AMAST 1991, Iowa City, pp. 1–23. Springer, Heidelberg (1992)
58. Pratt, V.R.: Chu spaces: complementarity and uncertainty in rational mechanics. Technical report, TEMPUS Summer School, Budapest (July 1994) (manuscript), http://boole.stanford.edu/pub/bud.pdf
59. Pratt, V.R.: Time and information in sequential and concurrent computation. In: Ito, T. (ed.) TPPP 1994. LNCS, vol. 907, pp. 1–24. Springer, Heidelberg (1995)
60. Pratt, V.R.: Chu spaces and their interpretation as concurrent objects. In: van Leeuwen, J. (ed.) Computer Science Today: Recent Trends and Developments. LNCS, vol. 1000, pp. 392–405. Springer, Heidelberg (1995)
61. Pratt, V.R.: Types as processes, via Chu spaces, Santa Margherita. In: Electronic Notes in Theoretical Computer Science, Santa Margherita, vol. 7, p. 21 (1997), http://www.elsevier.nl/locate/entcs/volume7.html
62. Pratt, V.R.: Chu spaces: Notes for school on category theory and applications. Technical report, University of Coimbra, Coimbra, Portugal (July 1999) (manuscript) http://boole.stanford.edu/pub/coimbra.pdf

63. Pratt, V.R.: Higher dimensional automata revisited. Math. Structures in Comp. Sci. 10, 525–548 (2000)
64. Pratt, V.R.: Orthocurrence as both interaction and observation. In: Rodriguez, R., Anger, F. (eds.) Proc. Workshop on Spatial and Temporal Reasoning, IJCAI 2001, Seattle (August 2001)
65. Pratt, V.R.: Event-state duality: The enriched case. In: Brim, L., Jančar, P., Křetínský, M., Kučera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, p. 41. Springer, Heidelberg (2002)
66. Pratt, V.R.: Chu spaces as a semantic bridge between linear logic and mathematics. Theoretical Computer Science 294(3), 439–471 (2003); Selected papers from Linear Logic 1996, Tokyo
67. Pratt, V.R.: Transition and cancellation in concurrency and branching time. Math. Structures in Comp. Sci., Special Issue on the Difference Between Sequentiality and Concurrency 13(4), 485–529 (2003)
68. Riddle, W.: The Modeling and Analysis of Supervisory Systems. PhD thesis, Computer Science Dept., Stanford University, p. 174 (March 1972)
69. Rodriguez, R.V., Anger, F.D.: Branching time via Chu spaces. In: Rodriguez, R., Anger, F. (eds.) Proc. Workshop on Spatial and Temporal Reasoning, IJCAI 2001, Seattle (August 2001)
70. Sassone, V., Cattani, G.L.: Higher-dimensional transition systems. In: Proceedings of LICS 1996 (1996)
71. Shields, M.: Deterministic asynchronous automata. In: Neuhold, E.J., Chroust, G. (eds.) Formal Models in Programming. Elsevier Science Publishers, B.V., North Holland (1985)
72. Takayama, Y.: Extraction of concurrent processes from higher-dimensional automata. In: Kirchner, H. (ed.) CAAP 1996. LNCS, vol. 1059, pp. 72–85. Springer, Heidelberg (1996)
73. van Glabbeek, R.: Comparative Concurrency Semantics and Refinement of Actions. PhD thesis, Vrije Universiteit te Amsterdam (May 1990)
74. van Glabbeek, R.: Bisimulations for higher dimensional automata (June 1991) (manuscript), `http://theory.stanford.edu/ rvg/hda`
75. van Glabbeek, R.: On the expressiveness of higher dimensional automata. Theoretical Computer Science 356(3), 169–194 (2006)
76. Winskel, G.: Events in Computation. PhD thesis, Dept. of Computer Science, University of Edinburgh (1980)
77. Winskel, G.: Event structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 255. Springer, Heidelberg (1987)
78. Winskel, G.: An introduction to event structures. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1988. LNCS, vol. 354. Springer, Heidelberg (1989)

Jump-Start Cloud: Efficient Deployment Framework for Large-Scale Cloud Applications

Xiaoxin Wu[1], Zhiming Shen[2], Ryan Wu[3], and Yunfeng Lin[4]

[1] Huawei Corporate Research, China
xiaoxinwu@huawei.com
[2] Department of Computer Science, North Carolina State University, US
zshen5@ncsu.edu
[3] tuan800.com, Beijing, China
ryanwu510@hotmail.com
[4] Intel China Research Center Ltd., Beijing, China
yunfeng.lin@intel.com

Abstract. [1]Reducing the time that a user has to occupy resources for completing cloud tasks can improve cloud efficiency and lower user cost. Such a time, called *cloud time*, consists of cloud deployment time and application running time. In this work we design *jump-start cloud*, under which an efficient cloud deployment scheme is proposed for minimizing cloud time. In particular, VM cloning based on disk image sharing has been implemented for fast VM and application deployment. For applications with heavy disk visits, the post-deployment quality of service (QoS) may suffer from image sharing and consequently, application running time will increase. To solve this problem, different image distribution schemes have been designed. We test jump-start cloud through a Hadoop based benchmark and MapReduce applications. Experiment studies show that our design saves application installation time and meanwhile, keeps application running time reasonably low, thus makes cloud time shorter.

1 Introduction

Cloud [1] [2] [3] has been looked as a natural evolvement for data center (DC) so that resources such as CPU, memory, storage, and IO/network in a DC can be dynamically and flexibly grouped or allocated, to serve clients with different service level agreement (SLA). Virtualization [4] [5] [6] [7] [8] [9] has been looked as a de facto management technology because of the speed, flexibility and agility it brings to cloud resource management. Providing resources in terms of virtual machines (VMs), it is easy to assign an application with a set of quantitatively measurable resources, e.g., a number of VMs where each VM has an assigned CPU slots (VCPU) and an amount of memory. As applications and OS are bundled into VM images, virtualization enables users to run their applications with a supporting OS of their choice, giving users the exposure to system-level

[1] This work was mainly conducted when Xiaoxin Wu, Zhiming Shen, and Ryan Wu were with Intel China Research Center.

capabilities, as provided by OpenCirrus open cloud testbed [10]. This is oppose to applications that only get access to higher level programming primitives as specified by Google AppEngine [11] and Microsoft Windows Azure [12].

Deploying and launching a VM-based user cloud application on top of a cloud infrastructure in general involves the following steps. First, a VM image mainly consisted of the targeted OS and user applications is compiled and then uploaded to a VM image repository within the cloud infrastructure. Then a number of VM's are launched on a set of DC physical hosts. Finally, application environments (IP, hostname, etc.) are configured and target applications are deployed.

It is desired that a cloud can *jump-start*, i.e., a cloud can be deployed and ready in short time. We think *cloud jump-start time*, in this work defined as the overall time that a cloud application is deployed and becomes functioning under desired QoS, should be fast for improving cloud infrastructure services for the following reasons:

1. One of the most important goals for cloud service providers is to make quick response to clients' resource requests and provide clients with applications running in desired states as soon as possible.
2. Even during the deployment time the resources reserved for a cloud application cannot be used by any other applications, which means a kind of resource waste. Making clouds jump-start can significantly improve cloud resource efficiency because a cloud normally serves thousands of users.
3. Users are usually billed based on the amount of cloud resource requested and the length of time to occupy such resource. The overall time that a cloud application occupies the resources needed, which we call *cloud time*, then is an important criteria for measuring, e.g., non long-lasting cloud services. It is to the interests of users to reduce cloud jump-start time for minimizing cloud time, unless the prompt cloud deployment results in a much longer application execution time.
4. Reducing cloud jump-start time helps improve cloud service availability. Various system software and hardware failures or software patches may require the entire cloud to shut down and restart. Faster jump-start time will help reduce system downtime and improve cloud availability.

Reducing cloud jump-start time has become a research challenge, especially for virtual cloud environment that normally involves a distributed system consisted of a large number of VMs. Having a virtual cloud environment ready requires many VMs deployed with a complex configuration setting, thus leading to challenges for both VM deployment and application environment setup. For example, it takes Amazon Elastic MapReduce [13] about 4 minutes to deploy a Hadoop cluster [14] with 20 VMs.

We propose using VM cloning to reduce VM installation time, thus to achieve a shorter jump-start time. A small-sized memory state file is generated from a suspending VM and distributed. Upon receiving the memory state file a VM can be resumed in a short time. The base image is shared in an image server, e.g. a NFS server, and VMs fetch extra data from the base image on-demand, e.g., through leveraging remote disk image access techniques that have been widely used in VM migrations [15] [16] [17] [18] [19].

A design issue not clearly addressed by previous cloning works such as [20] [21] [22], is how to differentiate cloned VMs. cloning from the memory state can only generate identical VMs. These VMs can not start working immediately before their memory states for MAC/IP addresses and the roles they play in the cloud application are assigned.

Another design issue not addressed previously, which we think even more important, is how to guarantee application post-deployment QoS. The QoS may suffer from VM cloning when there are a large number of live VMs sharing the same disk image, and for each VM disk image access is frequent. The disk IO for accessing the image then can be crammed, which leads to a long delay for image data fetch. The delay may cause application interruption and bad user experience for real-time applications, or a longer task accomplishment time for result-oriented applications.

We design and evaluate a comprehensive jump-start cloud deployment framework that considers all of the above issues. We implement VM cloning for fast VM launching, design specific dameon for VM differentiation and the consequent application deployment, and propose different disk image distribution methods for meeting post-deployment cloud QoS. We focus on non long-lasting VM-based Hadoop applications, and consider major QoS as how soon a cloud task is accomplished. The evaluation criteria then becomes cloud time, which is composed of jump-start time and application running time. The research methodology, however, can be applied to exploring how the proposed framework works for long-lasting applications, by investigating intermediate stages of applications instead.

In summary, our major contributions are as follows:

– We design jump-start cloud deployment framework that balances deployment time and post-deployment QoS. In this framework a KVM fast cloud application installation scheme is a designed. The scheme clones VMs upon memory state file, differentiates VMs through post VM configurations, and installs application through VM meta-data. Depending on application image access patterns, different image distribution schemes are designed to solve image access bottleneck problem.
– We identify cloud jump-start time as a critical parameter for cloud service measurement, and propose using cloud time that consists of cloud jump-start time and application running time for measuring non long-lasting applications.
– We design a Hadoop [14] cloud application benchmark. Through running it and cloud applications, we conduct intensive experiments for understanding, evaluating, and validating the proposed schemes. Experimental data shows that the proposed framework minimizes cloud time by making the time consumed on both application installation and application execution relatively short.

2 Jump-Start Cloud Deployment Overview

The targeted usage is user-defined large-scale cloud applications where both applications and their execution environment (e.g., OS) are created and/or defined by client. This may likely be one of major cloud usage models because future cloud is able to expose clients its infrastructure and let client determine the software environment. When the image is created by client, cloud services may have 1) a better efficiency due to a close bundle between application and its execution environment and 2) a better privacy because the client owns the entire software stack and data. In this model, to request cloud resources a client submits disk image to the cloud. The cloud works on this image to deploy and run the user-defined application. The application is on-the-fly, and for each request cloud has to establish a new execution environment. After the application is completed, this environment including VMs and data will be removed.

Our goal is to design a comprehensive cloud deployment framework that can make a non long-lasting user-defined cloud application finished in shortest time, i.e., makes its cloud time minimum. We care about the cloud time for result-oriented applications because for client a shorter completion time means a better QoS and a lower cost (a client will be charged based on a shorter resource occupation time), and for cloud a shorter cloud time can improve cloud resource efficiency (the released resources from the finished application can be used by other clients).

The two major steps after the cloud has loaded user image is application deployment and application running. To make the application deployment time (including the VM deployment time) short, our proposed technology for jump-start cloud is VM cloning. VM memory state files (or snap shots) are generated first at a physical machine where the application image is loaded. They are then delivered to the physical machines where VMs are supposed to be located at. As the size for a state file is small, the time for distributing it is short. VMs can then be quickly activated at destined machines. Once application meta data is sent to these VMs accordingly, application can start to run.

Such a fast application deployment may cause negative impact on application performance. A VM may visit the image from time to time. This happens, e.g., when disk image contains a large amount of data that has to be processed by the cloud application. Generally, a number of VMs will share and visit the same image, e.g., through NFS. In case a large number of image visits from a large number of VMs over the disk access IO, an image visit for a VM may take long time. If we consider the case that for an application the parallelism has been done perfectly at VM level so that there is only single thread in each of the VMs, the increased image access delay for a VM will result in the same increased running time for the application part running in that VM. Keeping image access delay low thus is critical.

In our efficient cloud deployment framework we add one more step before application deployment. The application disk image will be distributed to a number of physical machines within the cloud, if needed, to keep cloud time low for applications with a large number of image visits. Basically, a few image copies will serve VMs so that the number of VMs that share the same image becomes smaller. Consequently, the request rate for visiting a particular image becomes lower, which leads to a lower average image access delay. Because distributing image also takes time, whether or not to distribute images depends on applications' image visiting pattern, including application scale (number of VMs) and image visiting load per VM. In jump-start cloud images can be distributed before application is deployed or after application has already started.

3 Fast Cloud Application Deployment

Fast distributed application deployment in virtualized environment include VM installation and application deployment. In this section we present the detailed design and our solutions for implementation issues.

3.1 Fast VM Installation and Differentiation

For distributed cloud applications we deploy VMs through live cloning in a cloud that supports KVM. The memory state file of the original VM is distributed to the hosts (i.e., physical machines) where the VMs are supposed to be located. At this stage we consider all of VMs share one disk image, e.g., through NFS. Since the size of memory state file is small, distributing such a file within cloud takes a short time. Once a destined host receives the state file, the VM can be resumed at that host immediately. The state file can be either submitted by a cloud user or generated by cloud itself, through storing the memory state into a separate file. This is supported by mainstream hypervisors such as Xen [4] and KVM [5].

The cloned VMs are exactly the same as their parent, including networking configurations for IP address and MAC address. The next step for VM installation then is how to enable these VMs with individual network configurations and other distinguished features if any. Ideally, if a hypervisor at the host of that VM knows which part of the VM memory state should be changed for such diversification, it can modify the internal memory state and consequently diversify the VM. However, up-to-date there is no such technology due to the extreme complexity of memory state file.

In this work we diversify VMs by adding a daemon into each of them. The daemon will load the information needed, called VM-metadata in this paper, to distinguish its host VM. The VM-metadata is obtained from cloud manager. Based on the metadata the daemon triggers a re-configuration for the VM to have its distinguished features enabled.

A remained design issue is how a daemon obtains the metadata of a VM. Note that a daemon cannot obtain the metadata from the cloud manager directly through the DC network, as the VM cannot communicate with any other cloud components until its diversification is completed. Therefore, the metadata can only be obtained through some other ways based on the virtualized hardware.

In this work we collect VM metadata by generating an ISO image, as what has been done in VMPlants [23]. The metadata stored in the ISO image will be delivered to VM host along with the memory state file, while the daemon accesses the information from CDROM. The ISO image is customized through an interface open to cloud users. When creating a VM image the user adds additional information to VM-metadata, e.g., through a user script added to the ISO image. Once the ISO image is invoked after the VM cloning, customized configurations including application configurations will be accomplished.

We implement the fast VM deployment in KVM, thanks to the fact that KVM hypervisor supports offline migration, during which memory state is stored into a file and then loaded in the destination. Two important implementation works are VM meta-data construction and self-configuration daemon design.

3.2 Cloud Application Deployment

Once VMs are installed, each of them has to act as different roles of a distributed application. If a VM template also carries application memory state file, the application can be installed along with VM. However, in some applications such state file depends on cloud settings, which cannot be obtained before it starts to run in cloud. For example, in a process of HDFS in Hadoop, the storage ID is generated according to the cloud configurations and once it is set, it can not be modified. It is then difficult for a client to provide a storage ID in the VM template.

We extend the VM-metadata and design a framework that makes dynamic application configuration automatically. Under this framework, client adds a script to the VM-metadata that implements the client interface, and sends a script to a centralized configuration management server (CMS) that is designed for collecting information and delivering customized configurations. The CMS then can assign the VM a role through sending meta data, to initiate application deployment. After the fast installation of VMs the CMS has the system VM deployment information through collecting reports from these VMs. It then delivers configuration commands based on client's request, through which the role of each VM is determined. For example, when deploying a Hadoop cluster in the cloud, After receiving VM installation completion reports from all VMs, the CMS dbecides which VM works as Master. It then tells other VMs the Master's IP address and their roles, which are Slaves. Slaves then contact the master to get application started.

The application configuration framework is implemented with Python. When a request is submitted to the cloud stack, the CMS makes a new waiting list for VMs regarding to the request. After configuring network, the daemon in a VM will invoke the script named $selfconf.py$ in the VM-metadata. The activated VM then communicates with the CMS using remote procedure call (RPC). Upon receiving a PRC, the CMS invokes the user policy to make a decision what should be returned to the VM. The returned value can be additional meta-data or a configuration command. Typically there is only one interaction between a daemon and CMS. A daemon can trigger more RPC calls if more information is required.

4 Image Distribution for Application QoS

When applications depend on frequent, large number of disk image visits, the delay for a VM to obtain data from disk image may be large. This lowers computation efficiency and results in an increased application running time (i.e., a worse QoS). To reduce the average image access time, disk image can be distributed to a number of physical machines so that each of the images is visited by fewer VMs. In general, image distribution helps to reduce cloud time only if the time cost for the image distribution is less than the consequent time savings on disk image reading.

Since the size of an image is generally very large, it will take a long time to copy an image from one physical machine to another. In addition, for any part of a large-scale application, which runs in a VM in our case, it may only visit a small portion of the image. It is then not necessary to distribute the whole image to every host. Another reason for not distributing too many images within cloud is for storage savings, as storage is considered as cloud cost as well.

Ideally, if client or Cloud is aware of which part of image a VM will visit, the image part that will be visited by a lot of VMs then can be properly distributed. Unfortunately, identifying roles of each part of an image and separating them is not an easy work. Therefore, as a first step, in this work we consider the case that client or Cloud does not have the detailed knowledge of the image contents. However, they may know whether the application will visit the image frequently or not either through previous experience or Cloud monitoring.

The major image distribution schemes we have implemented in jump-start cloud deployment framework are pre-deployment, background deployment, and on-demand deployment.

4.1 Pre-deployment

In the pre-deployment distribution scheme the image will be copied and distributed in cloud before application runs. Given a client request that has defined the number of VMs required for running the application, the cloud decides the number of images to be distributed so that for each image, the number of VMs it will serve is below a threshold value. The number of images to be distributed depends on image reading load from each of the VMs as well as the network and disk IO bandwidth availability.

After the number of disk images to be distributed in the Cloud has been determined, a remained issue for pre-deployment is where to store these images (i.e., where to locate these image servers) and how to store them. Image server location depends on cloud system deployment and configuration. A general guideline is to store an image to make it easily accessed by its serving VMs. In our framework, an image is located at the same subnet as its serving VMs. After receiving an image, the image server stores it in its local disk. Because reading from disk takes longer time, to enhances image access speed, the image can be stored in the memory. This, however, is at the cost of occupying a much larger size of memory.

For applications that client or cloud is aware which part of the image may be visited by applications, only that part needs to be distributed. A typical such application can be one that processes client-provided data, where data is the major disk part to be visited and distributed. Other disk parts, e.g., application execution OS, are shared by all the VMs, unless image access bottleneck problem occurs then we can use background deployment method introduced later to resolve it. Distributing partial image significantly reduces image distribution time.

4.2 Background Deployment

Usually for any submitted cloud application it is not easy for either client or cloud to predict image reading pattern. Therefore, it is hard to make decision whether to pre-deploy disk image and how many images should be deployed. In our framework, for applications without the knowledge of image visiting patterns, all VMs share a single image at the beginning. Through monitoring the image access load as well as image reading time, the Cloud manager then decides whether the image has to be copied and deployed at other physical machines. During image deployment, application may continue running.

The number of images that should be finally distributed in the cloud can be reached gradually by adding images from time to time, until image access time is below a required value. This value is estimated through observing under what access latency application performance may not be seriously affected.

An implementation issue regarding to background deployment is about the so-called VM re-direction. Because at the beginning all the VMs are directed to a single image, when other images are ready, some of VMs have to be re-directed to their new assigned image servers. In this work we resolve the issue by changing the image direction command in KVM, which determines the image access for VMs.

4.3 On-Demand Deployment

Lots of image parts, e.g., OS bootstrap and drivers, may not or seldom be read by applications. Therefore, it is not efficient to distribute the entire image. However, pre-deployment requires a clear knowledge for what part of an image will be accessed during application runtime, which is difficult. To address this issue, we propose on-demand image distribution scheme.

In this scheme, at the beginning a number of images are built along with the original image in cloud. These images are actually empty, and we call them pseudo images. Like the oringinal image (or real image), a pseudo image also serves a number of VMs. When a process running in a VM served by a pseudo image requires some image data, it will visit the pseudo image serving it. If the data requested is not available at the pseudo image, the pseudo image will ask the real image for that part. Once the pseudo image receives the data, it delivers it to the requesting VM. At the mean time, it stores a copy for serving other VMs that may require the same image part later. On-demand deployment may lead to a longer delay for some image readings, yet it can significantly reduce communication load for image distribution.

To further improve the performance of on-demand image deployment, multicast and data pre-fetching can be applied.

5 Experimental Results

5.1 Experimental Data

Hadoop Deployment Time. In Fig. 1 we show when applying our fast application deployment scheme that shares a single image, the deployment time of a Hadoop with different scales, i.e., different number of VMs. The overall deployment time is around 10 seconds, which is much shorter than deploying such a Hadoop in $EC2$. In addition, the deployment time does not significantly increase when Hadoop scales up, because in such application after loading the VM memory state file, a VM seldom visits the disk image. Access delay for disk reading at the image server then is short. The meta data delivered for VM role configuration has a small size too. Therefore, even if the number of VMs increases, there is a minor increase for application deployment time. In Fig. 2 we show the detailed time contribution from different stages of Hadoop deployment at each of the VMs, when 56 VMs are assigned. At this scale, the time for memory state file distribution (for VM cloning), VM resume, and application configuration are about $0.5s$, $2s$, $8s$ respectively.

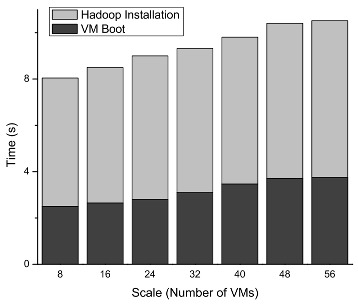


Fig. 1. Deployment time for Hadoop at different scales

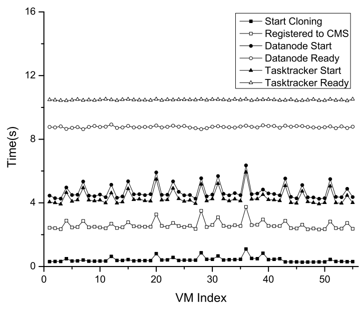


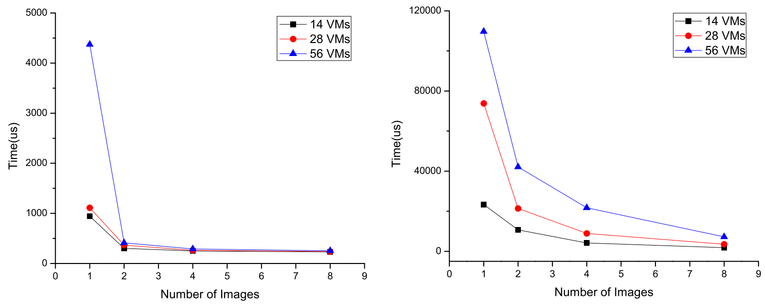
Fig. 2. Time taken at different stages and different VMs

In Table 1 we compare the time needed for deploying MapReduce with different scales in jump-start cloud and $EC2$. Generally, $EC2$ will take a few minutes while jump-start cloud takes less than 20 seconds. Considering a MapReduce application such as sorting may take only a few minutes, fast deployment through jump-start cloud can greatly improve user experience and cloud efficiency.

Table 1. Deployment Time: Jump-Start Cloud vs. $EC2$

Application Scale (Number of VMs)	20	50	100
Jump-Start cloud launching time	$13s$	$14s$	$15s$
EC2 launching time	$238s$	$291s$	$304s$

Improvement through Image Distribution. In Fig. 3 we shows distributing disk image does help to reduce average image access delay, and consequently the overall application running time. Under our testing scenario, when the number of images changes from 1 to 2, the delay decreases significantly. When the number of images increases further, the gain on access delay becomes trivial. This implies that for reducing access delay, distributing too many images may not be necessary.



(a) Sequential Access (b) Random Access

Fig. 3. Average access delay per request under different number of images

We show in Fig. 4 how image pre-deployment may help to reduce application cloud time when VM has a sequential reading pattern. Both unicast and multicast are used for image distribution for comparison. It is observed that image distribution works for heavy image visiting load. Multicast has a better gain because it helps to save image distribution time. From our experimental data we found that image pre-deployment can help to cloud time if when a single image serves all the VMs, its disk IO is fully occupied. As we mentioned previously, when VMs read disk sequentially, the equivalent disk IO bandwidth in our setting is about $20MB/s$ or $160Mb/s$. For the unicast case, if we use $1Gb/s$ for intra-cluster connection, the maximum number of images is approximately 2. The experimental results proves the mathematical deduction. When the number of images becomes greater, pre-deployment has a longer cloud time. This can be explained by referring to Fig. 3. A greater number of images results in very minor access delay improvement at a much longer image distribution time. The overall time saving on application execution then cannot compensate the time taken on distributing images. For the multicast image distribution case there is no such issue, through which generally the a greater number of images are distributed, a shorter cloud time can be achieved. However, when the number reaches a value, the gain from multicast is minor as well. Since the number of images to be multicasted determines how much cloud network bandwidth and system storage are needed, this number should be carefully selected based on application parameters and cloud configurations.

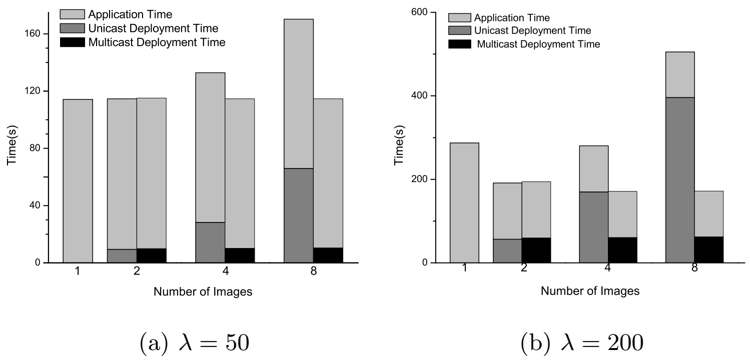
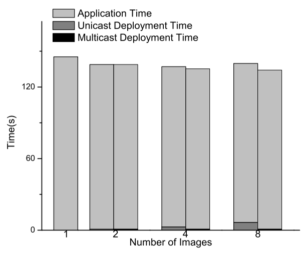


Fig. 4. Cloud time for application with sequential access under different number of images

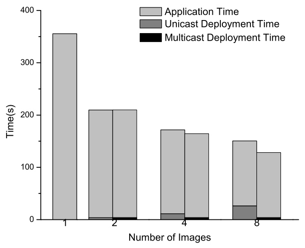
Fig. 5 shows the cloud time for random image access pattern. The overall performance trend is similar to what has been shown for the sequential disk reading case. For unicast the optimum number of image to be distributed, however, is greater than 2 because the equivalent disk IO bandwidth now is much smaller. Our test result shows the IO bandwidth is about $2MB/s$, the optimum number for distributing images is about 8.

Background and On-Demand Deployment. When using background image distribution we found there was no gain on cloud time reduction. The reason is that image distribution can help to reduce overall cloud time only when sharing single image, image disk IO is fully occupied. Therefore, background distribution under heavy disk visiting load will contend disk IO with applications, making the overall data getting out from disk image greater than the case when all VMs share a single image. However, background distribution may help is some particular application cases where QoS can be degraded to keep applications from being interrupted. For example, for a streaming video application at the beginning a fewer frames per second can be played, while the saved disk bandwidth can be used for image distribution. Once the background image distribution is completed, the performance becomes normal.

In Figure 6 we show cloud time improvement when on-demand image distribution is applied. Different numbers of images (including one real image) are deployed. We suppose 50% of the image will be accessed by all of the VMs,



(a) $\lambda = 5$



(b) $\lambda = 20$

Fig. 5. Cloud time for application with random access under different number of images

while other parts of the image will be accessed by VMs individually. It is observed that on-demand distribution can help to reduce cloud time. We also found that the deployment with 3 pseudo images has a little worse gain that that from 1 pseudo image case. The reason is that the visiting load to the real image is the performance bottleneck for on-demand distribution. This visiting load, actually, is determined by the number of the physical machines it serves. The number of physical machines the real image serves is greater in the 3 pseudo image case in our settings (The cloud has 8 physical machines), thus it has worse performance. It is also observed that deploying one pseudo image in the on-demand scheme results in a better performance than deploying one more real image through pre-deployment, because pre-deployment has to distribute the image part that will not be accessed as well. When the number of image increases, pre-deployment works better, at the cost of communication and storage.

In Figure. 7 we show how image access pattern impacts the cloud time gain in on-demand distribution. The access pattern we care about is how much of the image will be accessed (or shared) by all of the VMs. We consider only 1 pseudo image is deployed. According to the figure, it is observed that the more image data to be shared, the more cloud time reduction can be achieved through on-demand distribution. This is because the major role of the on-demand distribution is to reduce the visiting load at the real image. This load is mainly determined by the size of the disk data to be shared. Therefore, the on-demand distribution works better when more image part has to be shared.

Impact of Image Distribution on Real Application. In Fig. 8 a) we show the impact of image distribution on a MapReduce sorting application. There are $2G$ data to be sorted, through a MapReduce that runs over 56 VMs. The data is provided by user and originally stored in a disk image. We modified Hadoop so that data can be loaded from local storage. Each Mapper fetches the same amount data from the disk, and no data is shared. Mappers read data in turn, following a sequential reading pattern. As in our previous experiment, the tested results show the equivalent disk IO bandwidth can be up to $20MB/s$. Because

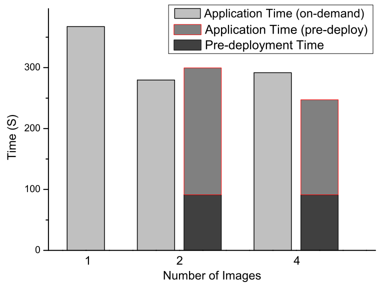


Fig. 6. Cloud time with on-demand image distribution

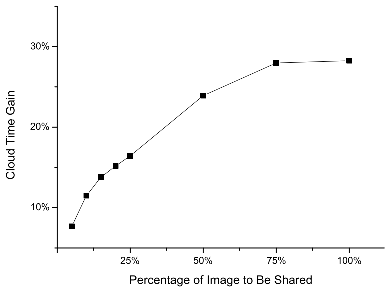
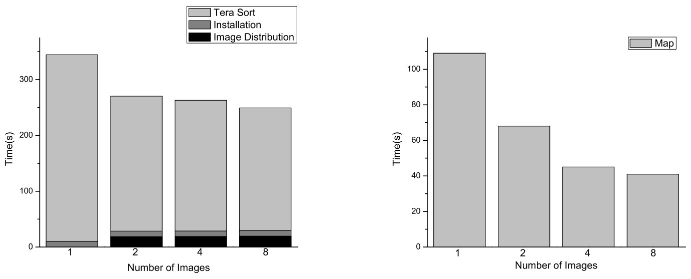


Fig. 7. On-demand distribution gain vs. image visiting pattern



(a) cloud time

(b) Map time

Fig. 8. Impact of image distribution on real application

the network has a bandwidth of $1G$, according to the analysis in 4.1 image distribution should be able to help to reduce cloud time. The experimental data proves it. We show in Fig. 8 b) that the major gain on application running time comes from Mapper, because that is the process involving most image access for loading data.

6 Conclusions

We enable jump-start cloud that applies an efficient deployment framework we designed to reduce cloud time for resource efficiency and service quality improvement. VM cloning is used for fast application deployment, and image distribution is used for post-deployment QoS. We test different application image access patterns and cloud system configurations to study cloud application characteristics and evaluate the deployment scheme.

References

1. Armbrust, M., et al.: Above the Clouds: A Berkeley View of Cloud Computing. Technical report, UC Berkeley Reliable Adaptive Distributed Systems Laboratory (2009)
2. EC2: Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/
3. Nimis, J., Tai, S., Sandholm, T.: What's Inside the Cloud? An Architectural Map of the Cloud Landscape. In: CLOUD (2009)
4. Barham, P., et al.: Xen and the Art of Virtualization. In: SOSP (2003)
5. Kvm, http://www.linux-kvm.org/page/Main_Page
6. Xenserver, http://www.citrix.com/English/ps2/products/feature.asp?contentID=1686939
7. Waldspurger, Carl A.: Memory Resource Management in VMware ESX Server. In: SIGOPS (2002)
8. Chase, J.S., Irwin, D.E., Grit, L.E., Moore, J.D., Sprenkle, S.E.: Dynamic Virtual Clusters in a Grid Site Manager. In: HPDC (2003)
9. Steinder, M., Whalley, I., Carrera, D., Gaweda, I., Chess, D.: Server Virtualization in Autonomic Management of Heterogeneous Workloads. In: Proc. 10th Integrated Network Management (IM) Conference (2007)
10. Open cirrus, https://opencirrus.org/
11. Google app engine, http://code.google.com/appengine/
12. Windows azure platform, http://www.microsoft.com/windowsazure/
13. Amazon elastic mapreduce, http://aws.amazon.com/elasticmapreduce/
14. hadoop, http://hadoop.apache.org/
15. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and Gray-box Strategies for Virtual Machine Migration. In: Proc. 4th Symposium on Networked Systems Design and Implementation, NSDI (2007)
16. Sapuntzakis, C.P., Chandra, R., Pfaff, B., Chow, J., Lam, M.S., Rosenblum, M.: Optimizing the Migration of Virtual Computers. In: Proc. 5th Symposium on Operating Systems Design and Implementation, OSDI (2002)
17. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy Live Migration of Virtual Machines. ACM SIGOPS Operating Systems Review (2009)
18. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (2005)
19. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Black-box and gray-box strategies for virtual machine migration. In: Proc. 4th Symposium on Networked Systems Design and Implementation, NSDI (2007)
20. Lagar-Cavilla, H.A., Whitney, J.A., Scannell, A.M., Patchin, P., Rumble, S.M., de Lara, E., Brudno, M., Satyanarayanan, M.: Snowflock: Rapid virtual machine cloning for cloud computing. In: EuroSys (2009)
21. Vrable, M., Chen, J., Ma, J., Moore, D., Vandekieft, E., Voelker, G., Snoeren, A., Savage, S.: Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. In: SOSP (2005)
22. Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N., Warfield, A.: Remus: High Availability via Asynchronous Virtual Machine Replication. In: NSDI (2008)
23. Krsul, I., Ganguly, A., Zhang, J., Fortes, J.A.B., Figueiredo, R.J.: VMPlants, Providing and Managing Virtual Machine Execution Environments for Grid Computing. In: SC (2004)

Capacity Estimation in HPC Systems: Simulation Approach

A. Anghelescu[1], R.B. Lenin[2], S. Ramaswamy[3], and K. Yoshigoe[4]

[1] Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322, USA
[2] Department of Mathematics, University of Central Arkansas, Conway, AR 72035, USA
[3] Industrial Software Systems, ABB Corporate Research, Bangalore 560048, India
[4] Department of Computer Science, University of Arkansas at Little Rock, Little Rock, AR 72204, USA

aanghel@emory.edu, rblenin@uca.edu, srini@acm.org, kxyoshigoe@ualr.edu

Abstract. As HPC (high performance computing) systems are extensively employed for heavy computational problems throughout heterogeneous environments, the scale and complexity of applications raises the issue of capacity planning. A cardinal aspect of efficiency is the job scheduler in any HPC systems. The job scheduling techniques can worsen or mitigate issues such as job starvation, increased queue time, and decreased system utilization. Since the impact of scheduling techniques is dependent on the workload of a supercomputer, this research proposes to analyze various scheduling disciplines on a given workload. By simulating HPC system, for any given workload, we can find the paradigm that yields the best performance, i.e. minimizing the wait time of jobs in the queue while maximizing resource utilization. Furthermore, given a fixed configuration of a HPC system, this research can be used to determine an appropriate workload that optimizes the system's performance. The development and implementation of such complex simulation framework for HPC does not yet exist in HPC's literature. The efficiency of the proposed simulation framework is illustrated through simulation results of performance measures such as average queuing time, average number of jobs in the queue, and system utilization. These results are verified by a developed mathematical model for job load characterization.

Keywords: queuing disciplines, job load characterization model, discrete event system simulation, average queuing time, utilization.

1 Introduction

As HPC (supercomputers) systems are extensively employed for heavy computational problems throughout heterogeneous environments, the scale and complexity of applications raises the issue of capacity planning. A cardinal aspect of efficiency is the job scheduler in any HPC systems. The job scheduling techniques can worsen or mitigate issues such as job starvation, increased queue

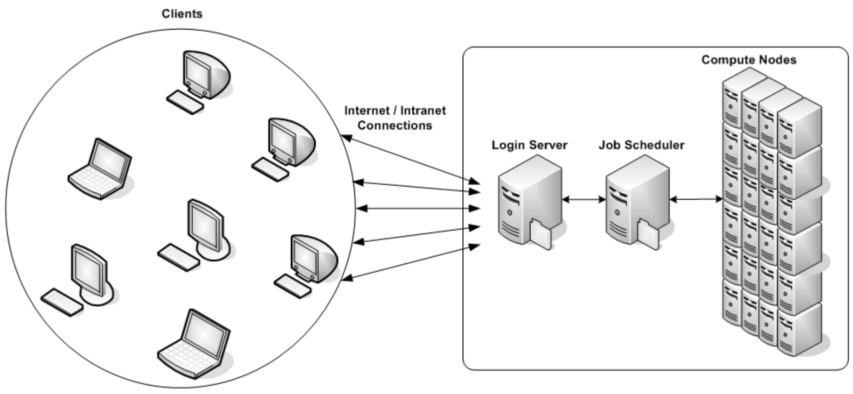


Fig. 1. High Performance Cluster Computing (HPCC) system overview

time, and decreased system utilization. Scheduling is a key concept in dealing with challenges such as capacity planning. The queuing algorithms implemented in the job scheduler can drastically change the performance of a supercomputer, in particular the CPU utilization and the wait time of a job. A software enhancement in the scheduler can obviate a hardware expansion by maximizing the utilization of the current system and improving its response time.

Figure 1 offers an overview of an HPC environment. Each job submitted allocates a number, n, of processors (from computing nodes) on which the job will run. The user decides this number when a job is submitted. It could be said that each job runs on its own partition of n processors [3]. An arriving job that does not find resources available to execute immediately will be placed in a queue. Wait time is the time that the job spends in the queue, and it varies depending on factors such as job priority, current load on the system, resources available, etc. [5]. Turnaround time is the total time elapsed between the submission and completion of a job; thus, it results from adding the wait time and the execution (processing) time. The queue is controlled by the supercomputer scheduler whose decision-making is based on the scheduling algorithm implemented; hence, various combinations of job parameters and scheduling algorithms yield different wait times and resource utilization percentages.

A brief survey of scheduling algorithms is presented in [5]. Even though most schedulers operate in space-mode sharing (jobs run on separate partitions which become available only after the job execution is completed), several scheduling algorithms are used in both commercial and open source schedulers. The most common and simple space-sharing algorithms are First In First Out (FIFO), Shortest Job First (SJF), and Longest Job First (LJF). FIFO executes jobs in the order in which they arrive in the queue. It is implemented easily, but wastes a lot of time and resources if the job load is high in the system. SJF and LJF periodically sort the incoming jobs in the queue and push the shortest, respectively longest jobs at the top of the queue. Obviously, SJF delays the execution of large jobs, while LJF yields a poor turnaround time. These algorithms can be enhanced by combining them with more advanced techniques such as backfill. The backfill technique improves space-sharing scheduling by trying to fit small jobs into gaps (i.e. idle compute nodes). When filling the gaps, the sequence of jobs previously scheduled is not altered. If job j which is at the top of the queue cannot start because the n processors that it asks for are not available, the scheduler looks back into the queue for the first job that could start executing with the current resources available. More scheduling algorithms or queuing techniques are discussed [2] and [13].

Reference [7] reveals that data gathered over 11 years of operating parallel supercomputers (including the Intel iPSC/860, Intel Paragon, Thinking Machines CM-5, IBM SP-2, and Cray Origin 2000) show three distinct trends: scheduling using the banal FCFS first-fit policy results in 40-60% utilization; employing more sophisticated dynamic-backfilling scheduling algorithms improves utilization by approximately 15% (resulting in 70% system utilization); and reducing the maximum allowable job size increases utilization. However the conclusion of [7] is that the goal of achieving 100% utilization is currently unrealistic and that the trends remain surprisingly consistent.

Advanced schedulers were developed to effectively operate in various HPC environments and efficiently adapt to the system's workload; two of the most popular and successful ones are dynP [12] and Maui [6]. The dynP scheduler dynamically changes three different queue disciplines (SJF, FCFS, and LJF), for incoming jobs, based on their job type characterizations. The implementation of a self-tuning scheduler with dynamic policy switching is continuously revised, as the boundaries of switching policies, and the fairness or unfairness of the decider, can be problematic. Maui [6] possesses an internal simulator that is capable of analyzing workload, resource and policy change impacts. In [5], Maui was touted as one of the better job schedulers because it possessed all the characteristics of a good job scheduler. Since the Maui job scheduler possesses an internal HPC system simulator, it is dependent on logs in order to maximize the scheduler performance. This means, it needs to have a running HPC system to make decisions.

Computer simulations have proven to be an excellent method of experiencing modifications to a real system in virtual time. Simulation has effectively been used to manage, evaluate and even predict a variety of systems including capacity planning and performance from mainframes to client servers [9], managing harbor container terminal [10], computer network reliability with congestion [8], and computer capacity planning [11]. In [4] the authors made an attempt to build a simulation framework for HPC systems using OMNeT++ 4.0. The workload for the simulation is modeled from trace log files of a HPC system. The simulation's purpose is to validate its results with the trace log files; however, key information is missing for the simulated HPC system, making the validation uncertain to some extent.

This research uses a simulation based approach to analyze scheduling techniques in HPC based on a given workload of the system. Using OMNeT++ 3.x, an extensible, modular, component-based C++ simulation library and framework [1], a HPC system will be simulated. In order to analyze the impact of scheduling techniques on performance, a varying configuration will be tested on a determined workload. The following four scheduling algorithms will be tested:

1. FIFO with backfill (no priority),
2. FIFO with priority and preemptive backfill,
3. SJF (no priority), and
4. LJF (no priority).

The impact on performance of the four scheduling algorithms will be measured by comparing the average wait times of jobs (queue time), the average number of jobs in the queue(queue length), resource utilization, and throughput(the ratio of jobs generated to jobs scheduled or completed). The strong aspect about this simulation is that the setup is dynamic. The results will show not only the present state of the system, but what happens as the configuration of the system is changed (i.e., changing the number of nodes, changing the policies). For systems that use a dynamic scheduler(switching between more policies), the simulation could decide when and what scheduling policies should be switched.

2 Job Load Characterization

This section presents the mathematical model for the job load characterization, which will help predict/verify the results from the simulations.

Let C and c denote the total number of clients, and the total number of CPUs of the HPC, respectively. For $i = 1, 2, \ldots, C$, let

- the random variable X_i^{ia} denote the inter-submission times of jobs submitted by client i to the HPC with mean $E[X_i^{ia}]$;
- the random variable X_i^{p} denote the processing time distribution of the jobs submitted by client i with mean $E[X_i^{p}]$;
- the random variable X_i^{cpu} denote the number of CPUs requested by jobs that are submitted by client i with mean $E[X_i^{cpu}]$;
- the ρ_i denote the average load of the HPC due to jobs submitted by client i.

By treating a job from client i asking for p CPUs as p jobs, the load ρ_i can be given by

$$\rho_i = \frac{E[X_i^{cpu}]E[X_i^{p}]}{cE[X_i^{ia}]}, \quad i = 1, 2, \ldots, C. \tag{1}$$

We note that $1/E[X_i^{ia}]$ gives the average submission rate of jobs of client i.

The total load ρ of the HPC offered by jobs of all the clients is then given by

$$\rho = \sum_{i=1}^{C} \rho_i = \frac{1}{c} \sum_{i=1}^{C} \frac{E[X_i^{cpu}]E[X_i^{p}]}{E[X_i^{ia}]}. \tag{2}$$

We note that ρ is independent of the queuing discipline adapted by the job scheduler.

3 Simulation Setup

OMNeT++ is used to build and run the HPC simulation software. OMNeT++ is an open source, extensible, and highly modular C++ simulation library and framework. OMNeT++ is gaining popularity as a simulation platform in the scientific community, and has a rather large user group. It is supported on all platforms, provides a rather thorough documentation, and generates a GUI for simulation execution. The GUI can be useful for visualizing the processes and debugging. A quintessential aspect of OMNeT++ is that the architecture is based on components, which are programmed in C++ and then assembled into modules using a high-level language(NED). It is primarly used for building network simulations, but we took advantage of its modularity and available libraries, and applied them to HPC. Because it is not specifically intended for HPC simulations, one of the challanges in using OMNeT++ was to adapt it to our project. The powerful simulation libraries and C++ were really important for our design. We currently implemented a very simple configuration for the HPC simulator, but for further development, OMNeT++ supports additional complexity such as switching, channel delays, latency, etc.

To make the simulation, the HPC system is abstracted into three modules. The client module, the job scheduler, and the node module. The client module is an array of individual clients which simulates the workload of the system. Each client has its own characteristic parameters such as job generation rate, processing time, range of nodes requested, any priority assignment associated with the job, and an unique ID. To simulate the workload we used distributions such as exponential for the job generation rate and processing times, and discrete uniform distribution for priority assignment and node request.

The job scheduler module encompasses the job queue, but also deals with communicating with the clients and nodes, and monitors the status of the compute nodes (busy or idle). The implementation of the job queue depends on the queuing discipline chosen. The first is FIFO with backfill, the second algorithm is FIFO with backfill and priority. In this case, each client must assign priority to its job. High-priority jobs can prevent lower-priority jobs from running if resources to run the high-priority jobs are not available. In some cases, resources reserved for high-priority jobs can be used to run low-priority jobs when no high-priority jobs are in the queue. This scheduling algorithm is sometimes called preemptive backfilling. The other two scheduling disciplines are SJF and LJF, where the job is sorted according to the number of nodes it is requesting, at insertion time in the job scheduler's queue.

The node module is an array of nodes, which represents the computing nodes of the HPC. For simplicity purposes, each node represents exactly one CPU. Statistics concerning the nodes utilization are collected in the scheduler module, and wait time is collected in the client module. The wait time is collected when the job gets out of the wait queue and it is sent to the nodes. Note that the wait time of the jobs that are still in the queue when the simulation ends will not be collected.

Each discipline will be tested on the simulated HPC system with a dynamic number of nodes. Overall there are 32 runs for each discipline. Run 1 is a HPC system with 32 nodes, run 2 with 64 nodes, run 3 with 96 nodes, run 32, respectively, with 1024 nodes. By varying the number of nodes, we can better analyze the tradeoff between wait time and resource utilization, and see the impact of the scheduling techniques on performance. There are 10 clients (this number stays fixed), each client having a different workload configuration from the others. Each run is five minutes long; hence, each simulation will have 32 runs of 5 minutes (wall clock time; in simulation time it can be days, months, or years) each.

In the simulation, we consider 10 clients and hence $C = 10$. The client i submits jobs according to an exponential distribution with an average inter-submission time of t_i^{ia} minutes. Each job of client i requests a number of CPUs based on a integer (discrete) uniform distribution over (a_i, b_i) with $a_i < b_i$. The processing times of jobs of client c_i is exponentially distributed with an average processing time t_i^p minutes. Therefore, X_i^{ia} is an exponential random variable with $E[X_i^{ia}] = t_i^{ia}$, X_i^{cpu} is an integer uniform random variable with $E[X_i^{cpu}] = \frac{a_i+b_i}{2}$, and X_i^p is an exponential random variable with $E[X_i^p] = t_i^p$. Hence by (1), we have

$$\rho_i = \frac{t_i^p(a_i + b_i)}{2ct_i^{ia}}, \quad i = 1, 2, \ldots, 10, \tag{3}$$

and by (2), we have,

$$\rho = \frac{1}{2c} \sum_{i=1}^{10} \frac{t_i^p(a_i + b_i)}{t_i^{ia}}. \tag{4}$$

The values of the parameters t_i^{ia}, t_i^p, a_i and b_i that are used in the simulation are tabulated in Table 1. Using these values, the total load on the HPC by all the clients is given by

$$\rho = \sum_{i=1}^{10} \rho_i = \frac{510}{c} = \frac{510}{c} \times 100\%.$$

That is,

$$\rho \begin{cases} \geq 100\%, \text{ if } c \leq 510, \\ < 100\%, \text{ if } c > 510. \end{cases}$$

Hence, theoretically in all the simulation results, after 510 CPUs, the load of the HPC drops below 100% and hence the average queuing time W_q and average number of jobs L_q in the queue should drop to zero and ultimately become zero as c increases. For the same reason, the throughput should reach 1 because all arriving jobs are processed because the system is under loaded for $c > 510$. This theoretically proven fact is exhibited by all our simulation results in Section 4 which validates the developed simulation framework's efficiency. We recollect that the load ρ of the HPC is independent of the queuing disciplines used by the job scheduler.

Table 1. Parameter values used in the simulation

Client i	t_i^{ia} (minutes)	(a_i, b_i)	t_i^p (minutes)	ρ_i
1	2.0	$(1, 5)$	4.0	$6/c$
2	4.0	$(6, 10)$	8.0	$16/c$
3	6.0	$(11, 15)$	12.0	$26/c$
4	8.0	$(16, 20)$	16.0	$36/c$
5	10.0	$(21, 25)$	20.0	$46/c$
6	12.0	$(26, 30)$	24.0	$56/c$
7	14.0	$(31, 35)$	28.0	$66/c$
8	16.0	$(36, 40)$	32.0	$76/c$
9	18.0	$(41, 45)$	36.0	$86/c$
10	20.0	$(46, 50)$	40.0	$96/c$

4 Simulation Results

In this section, we discuss the results generated from the output of the simulations. The graphs reflect the average waiting time W_q in the queue, job throughput T, the average length L_q of the queue, and system utilization.

Figure 2 illustrates the wait time of jobs in the queue, for each discipline. As it was mentioned in the previous section, it is important to note that the wait time is collected when the jobs get out of the queue. Thus, if the jobs never get out of the queue by the time the simulation ended, the time that they spend in the queue will not be counted. With this in mind, we plotted the throughput, Figure 3, and the average number of jobs in the queue, Figure 4, for each discipline. If the wait time is low, but the throughput is also low(hence, the length of the queue is relatively high), the wait time should be, in fact, very high. This is the case of clients 5-10, in Figure 2(a) for the node range 32-160.

Observing the graph in Figure 2(a), the wait time for the FIFO with backfill, we can see that the smaller the job is, the lower the wait time. Because this discipline uses backfilling, the smaller jobs are always favored, while the larger ones suffer, which was intuitive. In Figure 2(b) we introduce priority, which makes the results far less intuitive. Since priority is assigned uniformly for each job, the results are more balanced. Although backfill is still implemented, the smaller jobs are regulated by priority, and thus, will be prevented from jumping ahead of higher priority jobs. In this case, only clients 7-10 are completely blocked in the queue, but only for the configuration with 32 nodes (some clients asking for more nodes than are initially available). Introducing priority slightly increases the wait time of small jobs, but drastically decreases the wait time of larger jobs. Hence, on average, the overall wait time of clients in the module that uses uniform priority(Figure 2(b)) is much less than the one without (Figure 2(a)).

Figures 2(c) and 2(d) depict the scenarios for SJF and LJF. In Figure 2(c), since the jobs are sorted at insertion, the small incoming jobs will jump ahead of larger ones, and continually use up the resources. This results in job starvation for the larger jobs. For LJF it is vice versa. The wait time is so low for most

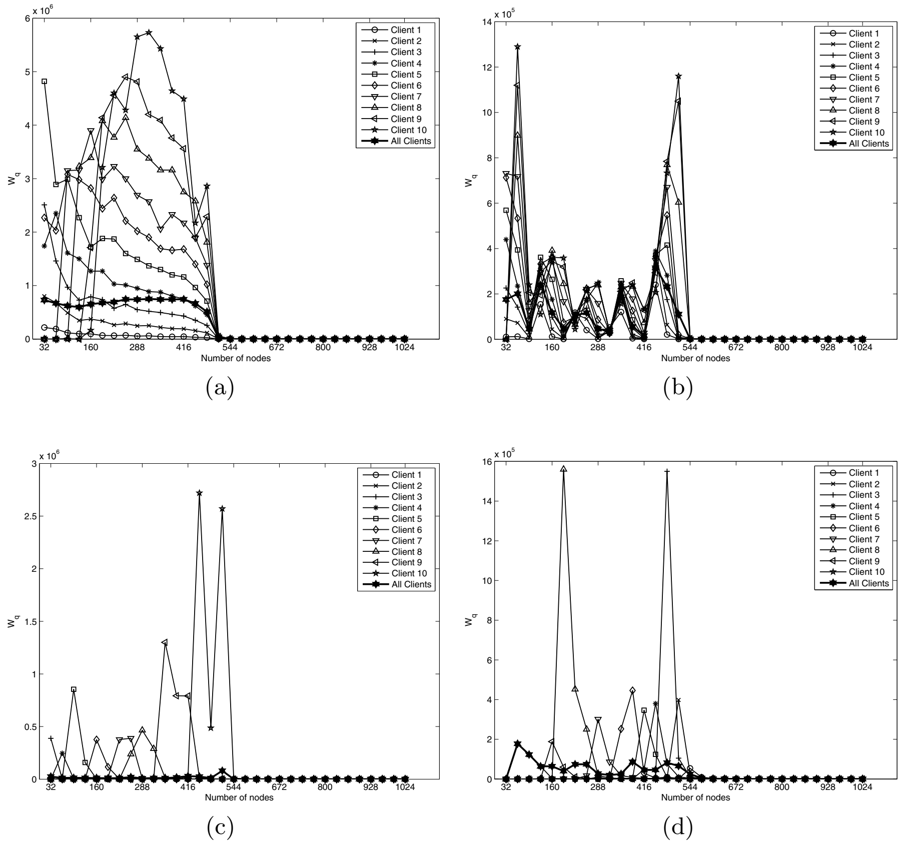


Fig. 2. Average wait time in the queue for each discipline

clients in SJF and LJF because, because the larger, and respectively smaller, jobs never get out of the queue. It is evident however, that around 500 nodes, the wait time of all disciplines approaches 0 (the system resources have outgrown the load on the system).

Analyzing the throughput of jobs in Figure 3 reveals the behavior of jobs in each discipline. In Figure 3(a) small jobs start out with a good throughput, while large jobs have 0 throughput. The smaller the client, the better throughput. This trend remains consistent for FIFO with backfill, and it is also the case of SJF (Figure 3(c)). However, in FIFO with backfill, all jobs reach a better throughput faster, and it is, thus, a more efficient policy for the given workload. LJF follows the reverse trend of SJF and FIFO with backfill - the larger the jobs, the better the throughput. This policy is evidently inefficient for this configuration of the system (large jobs are not predominant) since half of the clients (1-5) have a throughput of 0, even at 300 nodes. The throughput for FIFO with backfill and priority (Figure 3(b)) is significantly different than the others. It is clear here how the uniform priority balances the job scheduling. All job types have similar

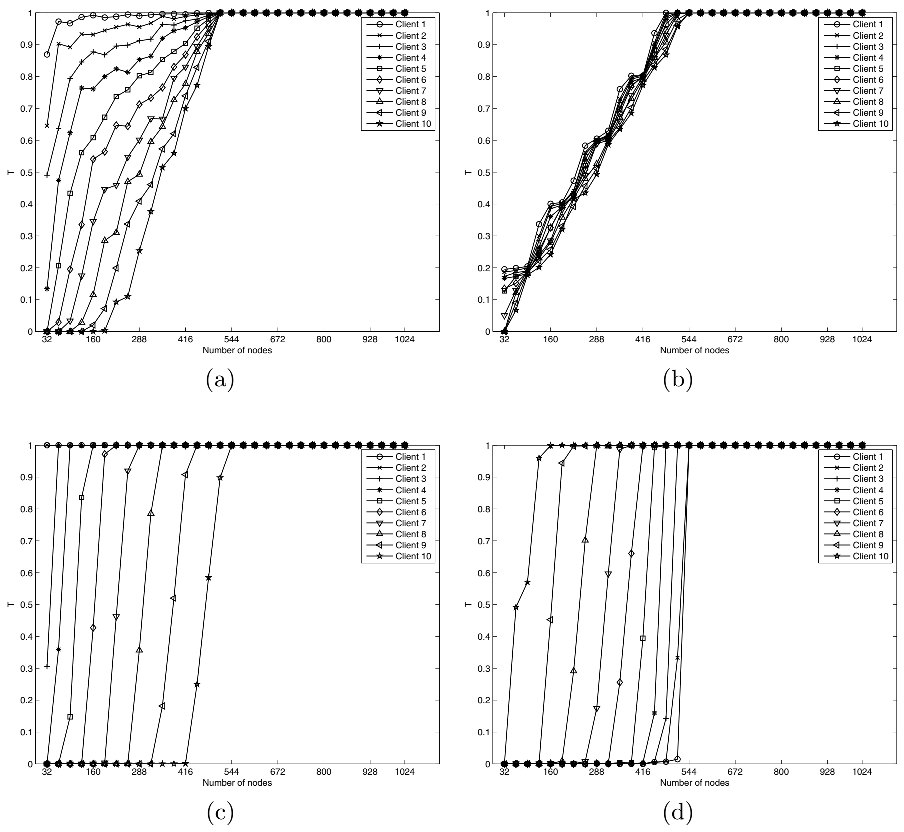


Fig. 3. Throughput for each discipline

throughput during the simulation. Around 500 nodes, all jobs reach a throughput of 1, which means that there are now enough resources available for all jobs to be completed.

Figure 4 shows the average length of the queue, for all job disciplines. This helps to visualize what is happening in the queue for each discipline. It is important to keep in mind that the smaller clients have a higher job generation rate than the larger clients. Actually, the job generation rate decreases as the jobs get larger. Figure 4(b) reveals that the length of the queue is larger for smaller clients. Since we know that all jobs in this discipline have similar throughput, the larger queue length for smaller clients must mean that there are more smaller jobs being generated, thus currently in the system, than larger jobs. In Figure 4(a), the smaller jobs have a shorter queue because the discipline favors them, and even though they are generated faster, they are also scheduled almost immediately. This tendency is present also for SJF in Figure 4(c), and the reverse for LJF in Figure 4(d).

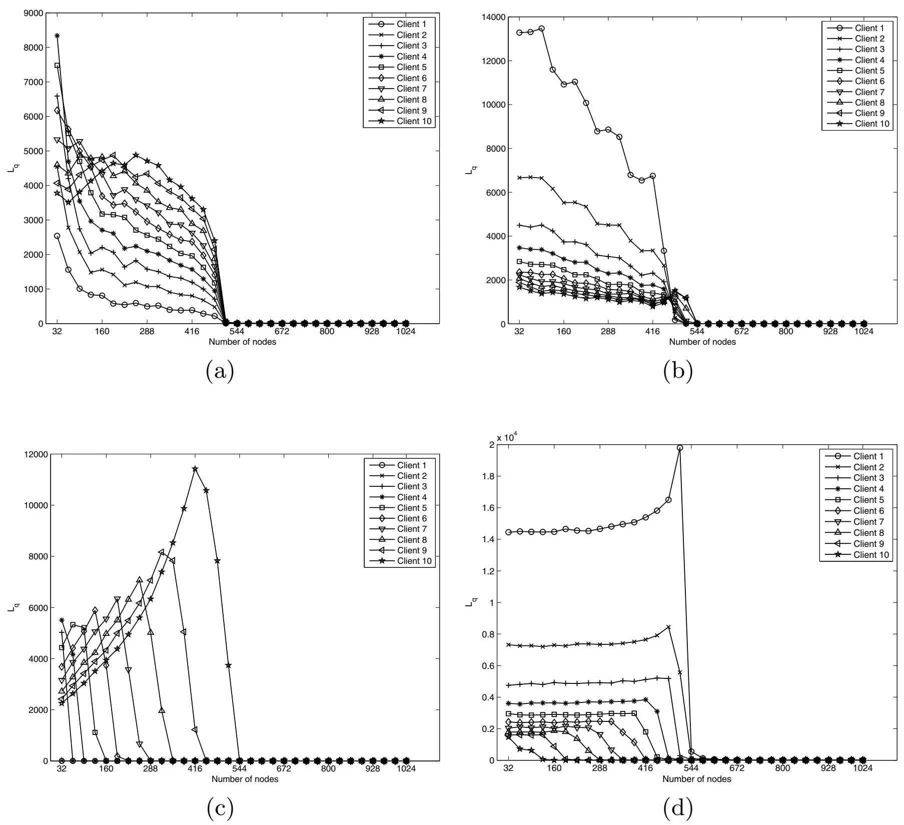


Fig. 4. Average number of jobs in the queue for each discipline

It is obvious that all four disciplines, for all the three performance measures, are convergent around 500 nodes. The wait time and queue length are rapidly decreasing to 0, and the throughput reaches 1. This reveals consistency, because a wait time of 0 means that the jobs generated are basically scheduled immediately, which means that the queue length is close to 0, and the throughput is 1 (every job generated is scheduled or completed during the simulation). Furthermore, the utilization graph in Figure 5 also shows that around 500 nodes, the resource utilization begins to drop exponentially. Before 500 nodes, the resource utilization for SJF and Priority is somewhat lower (90%) than FIFO with backfill (100%). The utilization for LJF starts high because for low resources, the large jobs fill up all the resources. But as the number of nodes increases, the utilization of the system is lower because the large jobs are blocking access to resources. In this case, some jobs are too large to be scheduled with the available nodes, and they don't allow smaller jobs to fill in. Utilization increases again when enough resources permit all the large jobs to start and allow smaller ones to start as well.

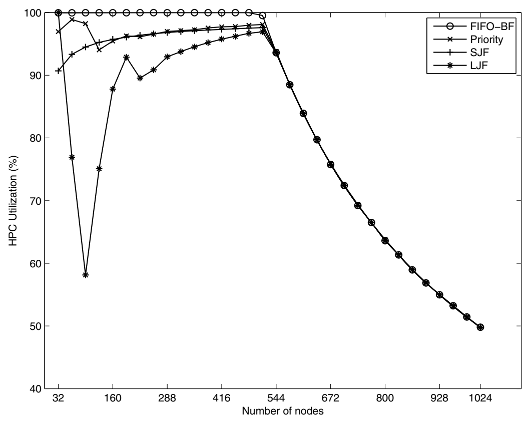


Fig. 5. HPC utilization

5 Conclusions and Future Work

The results of the simulation show how each queuing technique impacts performance and resource utilization. For the given configuration, it seems that FIFO with backfill and priority optimizes the system performance, yielding the best wait time and a moderate throughput, while utilizing resources in a better or similar way than other policies. The average wait time for SJF and LJF is thrown off by the numerous jobs that have a wait time of 0, when in fact, the wait time should be much higher. LJF is by far, the worst fit for the system. FIFO with backfill and SJF run close, but SJF offers too much favoritism to smaller jobs.

As the simulation results show, and the mathematical model validates, after 510 nodes, the resources outgrown the load of the system. This information can be useful for capacity planning purposes and hardware expansion. For the given workload trend, the 500-600 nodes seems an optimal choice for resource configuration.

These results only offer an insight into the capabilities of such complex simulation framework. The simulation can be used with any system workload that is modeled mathematically, and can be modified to fit various hardware configurations. For random workload configurations the simulation can be validated mathematically, and for a real workload configurations, the simulation can be validated by the logs that were used to model the workload.

Future work will include further analysis on the accuracy of the simulator. The goal is to compare the results obtained in the the simulation with ones obtained in the real environment. This can be addressed by configuring the simulator to be as close as possible to the HPC system that would be used to run the results, and modeling a workload from the HPC logs.

More work can be directed towards increasing the complexity of the scheduling algorithms (adding more parameters to the jobs such as threshold for the queuing time, memory limit, combining scheduling disciplines, etc.), and developing a comprehensive mathematical model to model an entire HPC system.

Acknowledgments

This work is based in part, upon research supported by the University of Central Arkansas, National Science Foundation (grant nos. CNS-0619069, EPS-0701890 and OISE 0729792), Grant CRI CNS0855248, Grant EPS0701890, Grant MRI CNS0619069, and OISE0729792. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

[1] OMNeT++ (2010), http://www.omnetpp.org
[2] Bansal, N., Harchol-Balter, M.: Analysis of srpt scheduling: Investigating unfairness. ACM SIGMETRICS Performance Evaluation Review 29(1), 279–290 (2001)
[3] Cirne, W., Berman, F.: Adaptive selection of partition size for supercomputer requests. In: Feitelson, D.G., Rudolph, L. (eds.) IPDPS-WS 2000 and JSSPP 2000. LNCS, vol. 1911, pp. 187–207. Springer, Heidelberg (2000)
[4] Hurst, W.B., Ramaswamy, S., Lenin, R.B., Hoffman, D.: Development of generalized hpc simulator. In: Proc. of Acxiom Laboratory for Applied Research 2010 (2010)
[5] Iqbal, S., Gupta, S.R., Fang, Y.-C.: Planning considerations for job scheduling in hpc clusters. Dell Power Solutions Magazine, 133–136 (February 2005)
[6] Jackson, D.B., Jackson, H.L., Snell, Q.O.: Simulation based HPC workload analysis. In: Proc. of International Parallel and Distributed Processing Symposium (2001)
[7] Jones, J.P., Nitzberg, B.: Scheduling for parallel supercomputing: A historical perspective of achievable utilization. In: Feitelson, D., Rudolph, L. (eds.) JSSPP 1999, IPPS-WS 1999, and SPDP-WS 1999. LNCS, vol. 1659, pp. 1–16. Springer, Heidelberg (1999)
[8] Lui, H.-L., Shooman, M.L.: Simulation of computer network reliability with congestion. In: Proc. of Annual Reliability and Maintainability Symposium, pp. 208–213 (1999)
[9] Menascé, D.A., Almeida, V.A.F., Dowdy, L.W.: Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems. Prentice-Hall, Upper Saddle River (1994)
[10] Merkuryev, Y., Tolujew, J., Blumel, E., Novitsky, L., Ginters, E., Viktorova, E., Merkuryeva, G., Pronins, J.: A modelling and simulation methodology for managing the riga harbour container terminal. Simulation 71(2), 84–95 (1998)
[11] Riesen, R.: Simulating a supercomputer. Presentation, Sandia National Laboratories, Wildhaus, Switzerland (March 2008), http://sos12.epfl.ch/riesen.pdf
[12] Streit, A.: The self-tuning dynp job-scheduler. In: Proc. of the 20th International Parallel and Distributed Processing Symposium, pp. 1530–2075 (2002)
[13] Thanalapati, T., Dandamudi, S.: An efficient adaptive scheduling scheme for distributed memory multicomputers. IEEE Transactions on Parallel and Distributed Systems 12(7), 758–768 (2001)

A Multi–Granular Lock Model for Distributed Object Oriented Databases Using Semantics

V. Geetha and N. Sreenath

Dept. of Information Technology, Dept. of Computer Science & Engg.,
Pondicherry Engineering College,
Puducherry – 605014
vgeetha@pec.edu, nsreenath@pec.edu

Abstract. In object oriented databases, transactions may make simultaneous requests to do design time access and runtime access of resources. Concurrency control on the transactions can be implemented by using Multi granular lock models. Though several semantics based multi granular lock models have been proposed in the literature for object-oriented databases, they provide fine granularity of resources for runtime requests. However they have not fully utilized the semantics of object-oriented concepts to provide fine granularity of design time requests. In the proposed semantic based multi granular lock model, various dependencies of objects are exploited to exercise concurrency control. The dependencies among objects participating in the system can be inferred through their relationships such as inheritance, composition etc, with other objects participating in the domain. The proposed lock model uses these dependencies in defining lock modes for design time requests, which will provide fine granularity. It also utilizes these dependencies to provide better concurrency control for transactions modifying class relationships using Relationship Vectors (RV).

Keywords: Distributed object oriented databases, concurrency control, multi granular lock model, class relationships, design time requests, run time requests.

1 Introduction

Object Oriented Database Systems (OODB) is a collection of classes and instances, where classes and instances are called objects. Multi- Granular Lock Model (MGLM) is a common technique for implementing concurrency control on transactions using the OODB.

There are several MGLM proposed in the literature. Gray et al. [1] has defined MGLM for relational databases. The main advantages of MGLM are high concurrency and minimal deadlocks. Using MGLM, transactions can request resources of different granule sizes varying from coarse granules to fine granules. Intension locks are used to infer the presence of locked resources at smaller granule level. The lock modes defined in [1] are S (Shared - Read), X (eXclusive – Write) and SIX (Shared Intension eXclusive – locks all in S mode but a few of them to be updated alone in X mode).

MGLM was first extended to object oriented databases by Garza and Kim [2] for ORION. In this paper, MGML was defined for objects related by inheritance and exclusive composition only. The locks defined in [2] are of granularity of classes (collection of objects) and objects. Later Kim et al. [3] extended it to all types of composition (namely shared and exclusive, dependent and independent). In this paper, apart from the lock modes in Garza and Kim [2], new lock modes like ISOS, IXOS, SIXOS are added to support shared composition. In Geetha and Sreenath [4], these shared intension locks are extended to shared inheritance also. In Jun and Gruenwald [5], concurrency control for runtime requests on inheritance alone is proposed. However, in [1, 2, 3, 4, 5] compatibility of runtime requests only are considered. A compatibility matrix defines the compatibility of a requested lock mode over existing lock mode on a particular resource.

Due to the continuous evolving nature of distributed systems, a system might receive both design time requests and runtime requests parallely. The compatibility is extended to consider design time requests also in Lee and Liou [6]. In [6], new locking modes like RS (Read Schema) and WS (Write Schema) are added to support design time requests. A new compatibility matrix has been designed in [6] to support both runtime requests and design time requests simultaneously. However, the design time request lock modes are defined in [6] for coarse grain i.e., entire schema only. The smallest granule in [1, 2, 3, 4, 5, 6] for runtime requests is only up to object level and all of them have proposed MGLM based on relationships only.

Malta and Martinez [7] proposed commutativity of methods to resolve lock conflicts between runtime requests. In [7], the lock modes are defined independent of object relationships. This paper has claimed to eliminate the burden of determining commutativity exhaustively for every pair of methods at run time, by determining it apriori using direct access vectors (DAV). A DAV is a vector defined for every method, whose field corresponds to each attribute defined in the class on which the method operates. Each value composing this vector denotes the most restrictive access mode used by the method when accessing the corresponding field. The access mode of any attribute can be one of the three values, N(null), R(read), W(write) with $N < R < W$ for their restrictiveness. The access vectors are defined for all methods based on their lock mode on every attribute defined in the class. This paper has also claimed to reduce locking overhead, lock escalation and deadlocks. Since the most restrictive lock mode is decided in the beginning itself, lock overheads due to lock conversions are reduced, and hence deadlock is minimized. However, this paper has extended concurrency up to attribute level.

In Jun [8], commutativity is further extended to a granularity lower than attribute level by using break points. Here the DAV of a method is computed as the union of DAV of all its break points, which improved the concurrency of the system further. The lock modes of design time requests defined in Jun [8] have been extended to three smaller granules for every class: - 1. Read/Modify Attributes, 2. Read/ Modify Methods, 3. Read/Modify Class Relationships. In Saha and Morrisey [9], a self adjusting MGLM is defined to let the transactions to dynamically choose their granularity from coarse to fine on a particular resource based on the increasing degree of resource contention.

Though [7, 8, 9] claim to be semantic based MGML, they have not really exploited the semantics of attributes, methods and class relationships to fully maximize the concurrency of design time requests. They perform better, when the system is stable with less frequent changes. This paper aims in proposing a MGML, which will improve the degree of concurrency for design time requests and runtime requests by fully utilizing the semantics of object oriented features.

The organization of the paper is as follows: Chapter 2 explains the contributions of the related papers. Chapter 3 explains the proposed MGML. Chapter 4 gives informal proof and chapter 5 concludes the paper.

2 Related Works

In this chapter, semantics related to various types of access conflicts are discussed. The resource access requests can be runtime requests or design time requests. A typical runtime request involves execution of methods in classes that will read or alter the values of attributes in the class. The values of the attributes are mapped on to the underlying database. The runtime access to a resource can be at class level (involving all objects) or instance level (involving any one object) based on the property of methods [11]. A design time request involves reading and modifying the structure of the system. Since it is OODBMS, the structure is defined by a set of related classes participating in the domain. The possible conflicts in resource access could be 1. Among runtime requests, 2. Among design time requests and 3. Between runtime requests and design time requests.

Since the granularity of runtime requests is already provided up to smallest granule i.e., attribute level, in Jun [8] by using DAV, we will focus on providing fine granularity for cases 2 and 3 in this paper.

2.1 Conflicts among Design Time Requests

In Garza and Kim [2] and Kim et al. [3], design time requests are not at all allowed. Lee and Liou [6] require locking the entire OODBMS schema for reading (RS) or writing (WS) by design time clients. This provides locking of coarse grains only and hence reduces concurrency. Malta and Martinez [7] defines lock modes RD (Read Definition) and MD (Modify Definition) to lock every class individually.

In Jun [8], Class definition has been divided into three compartments namely 1. *Reading and Modifying Attributes (RA, MA),* 2. *Reading and Modifying Methods (RM, MM) and* 3. *Reading and Modifying Class Relationships (RCR, MCR).*

Locking mode for attributes involves changing the domain of the attribute, or deleting the attribute. In Jun [8], there is only one lock mode shared by all the attributes of a class. At any time, only one attribute can be modified in a class. This lock mode considers the access conflicts within the class only. It does not consider the conflicts arising due to the relationship of this class with other classes. However, AAV (Attribute Access Vector) is defined for every attribute of all classes to maintain its lock status. Using this, simultaneous access to more than one attribute is facilitated. This paper offers a trade off between limited concurrency of accessing only one attribute at a time against maintenance overhead of AAV for all attributes of every class.

By the semantics of object-oriented concepts, Sub classes can both read as well as modify their attributes, but they can only read base class attributes. Only the base classes can modify the base class attributes. This theory can be extended to composition also. The attributes defined in component class can be modified only in component classes where as composite objects can only read them. In simple words, modification is possible only in the class in which the attributes are defined. Therefore, the attributes from the base classes and component classes can be viewed as *adapted attributes* in the sub class or composite class. So in Jun [8], adapted attributes and attributes defined in this class cannot be accessed parallely, even though they are mutually exclusive without AAV. It also does not define the concurrency control of modifying an attribute in base class (component class) while reading the same attribute in sub class (composite class).

In Jun [8], there is only one lock mode for all the methods defined in a class. At any time, only one method can be modified in a class. This lock mode considers the access conflicts of methods within the class only. It does not consider the access conflicts arising due to the inheritance and composition relationships of this class with other classes. However, MAV (Method Access Vector) is defined for every method to maintain its lock status. Using this, simultaneous access to all methods is facilitated. This paper offers a trade off between limited concurrency of accessing only one method at a time against maintenance overhead of MAV for all attributes of every class.

By object oriented semantics, modifying methods typically includes modifying the signature of the method, modifying its implementation and modifying its location i.e., moving a method from one class to another class in the class hierarchy. Modifying the signature means modifying the name of the method, adding or deleting the input parameters, changing their order and changing the returning type. It can be noted that modification of method is independent of reading and modification of attribute as they are to done by acquiring different lock modes and cannot be done at the same time. In Jun [8], all the operations related to modification of methods are locked in the same mode.

For example, consider figure 1. A1 is an attribute of base class and it is inherited in subclass. B1 is sub class attribute. M1 and M2 are methods of base class. In this M1 is inherited as it is in sub class (called as *template method* [10]), whereas M2 is overridden in subclass (called as *hook method* [10]). M3 is a method defined in subclass. By Jun [8], locking of all types of attributes i.e., A1 and B1 are defined by a single mode and locking of all methods i.e., M1, M2 and M3 are defined by a single mode in the subclass. The semantics of each of these attributes and methods are not utilized to maximize the concurrency.

There are certain aspects that can be inferred from fig 1. Attribute A1 can be read in both base class as well as sub class. However, modifying A1 is possible only in base class. It is worth noting that while base class is modifying A1, no request should be allowed in sub class to read A1 to maintain consistency. In sub class, attribute B1 can be read or modified. So the attributes in any class can be categorized into two categories namely, 1. attributes adapted from other classes and 2. attributes defined in the same class.

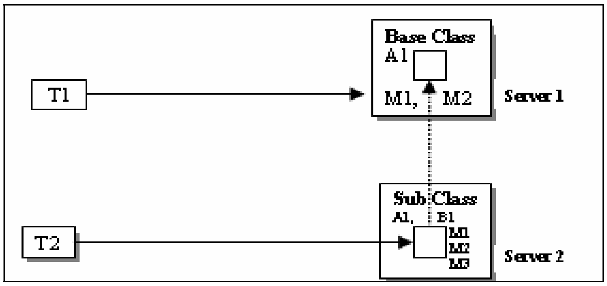


Fig. 1. Locking in Inheritance

In fig 1, M1 is a template method whose signature or implementation can be modified only in base class. It can be only read in subclasses. M2 is a hook method. Therefore, its signature is modifiable only in base class and implementation is modifiable in both base class and subclass. This means hook methods can be overloaded and can have different implementations in base class and sub class. In Jun [8], MM is the only mode to handle all these method types. Similarly, signature and implementation of methods defined in component class cannot be modified in composite class. They are available only for reading in the composite class. Hence, they can be treated similar to template methods of inheritance.

Modifying class relationship involves adding or deleting a class from the class hierarchy, moving its location in the class hierarchy. In Jun[8], class relationship definition includes name of the class, all attributes and methods defined in the class, set of super classes and sub classes of the class. This is not complete definition of class relationships. Banerjee et al. [13] categorizes modification of class relationships into two types: 1. changing an edge 2.changing a node.

Changing an edge means changing the relationships between any two classes in the class diagram. This includes making a class as parent class (component class) to a sub class (composite class), removing a class from the list of parents (component classes) of a class and changing the order of parent classes of a class.

Changing a node includes adding a new class, dropping an existing class and changing the name of a class. In Jun [8], at any time only one change in class relationship is allowed at a time in every class. It does not take into account the details relating between classes. I.e., it considers intra class relationships only and excludes inter class relationships. In [6, 7], no other runtime requests or design time requests are allowed. i.e., the entire class diagram is locked and indirectly the entire database is locked. So there is no lock mode defined in the literature to read or modify class relationships as defined in Banerjee et al. [13].

2.2 Conflicts between Runtime Requests and Design Time Requests

In object oriented databases, all objects in a class share the implementation of all methods defined in the class. Though storage space is allotted separately to attributes of all objects in a class, their method implementation is usually shared by all objects. Hence, it implies that while one or more objects simultaneously share the implementation of a method for reading and execution, if design time clients try to modify the implementation, the consistency of the system will be affected.

3 Proposed Scheme

The proposed scheme mainly aims in providing fine granule locking for design time requests. This will not only improve concurrency between design time requests, but will improve concurrency between runtime requests and design time requests also. The principles based on which concurrency is improved between design time requests and between runtime requests and design time requests are discussed in the following sections.

3.1 Concurrency among Design Time Requests

The proposed scheme defines the lock modes with following objectives:

1. Exploiting the features of object oriented systems to identify mutually exclusive operations in the system.
2. Maximize concurrency by providing rich set of locking modes.
3. Provide compatibility matrix independent of domain or specific instances, so that it does not require any apriori analysis.
4. Impose concurrency control wherever consistency is affected, due to semantics of object oriented concepts.

Based on section 2.1, the attributes are classified into Adapted Attributes (AA) and Attributes (A). Hence separate lock modes can be defined for reading adapted attributes(RAA) and reading and modifying sub class attributes (RA,MA). Since adapted attributes cannot be modified in this class, lock mode for modifying these adapted attributes is not available in this class.

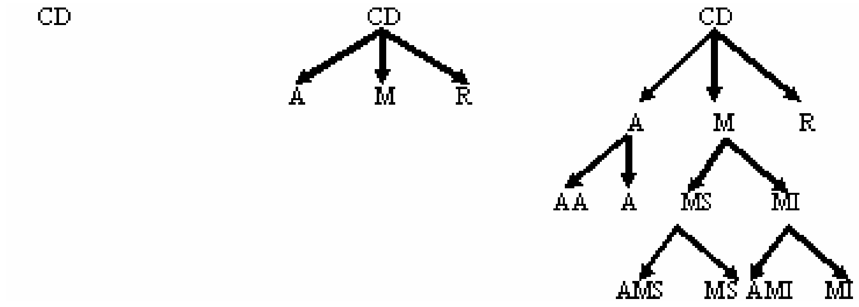
Szyperski [12] says that the signature i.e. method definition is independent of method implementation. This concept is called as separation of concerns. In object oriented environment, the implementation of a method can be modified any number of times. As long as its definition does not change, the clients need not be informed about the change in implementation. The implementation of methods is usually modified to provide better service to clients. However, when the signature is changed, the clients need to be informed, as they are going to avail this service only by calling in this format. In fact, the signature is viewed as a contract between client and server. So lock mode for method is split into Method Signature (MS) and Method Implementation (MI) in the proposed scheme.

Based on section 2.1, the signatures also called as definitions of methods in base classes and component classes are separately maintained in the class. Hence separate lock modes can be defined for reading signatures of adapted class methods (RAMS) and for reading and modifying sub class methods(RMS,MMS). Since signatures of methods in adapted class cannot be modified in this class, lock mode for modifying the signature of these adapted methods is not available in this class.

Modification of template methods of inheritance and methods in component classes is possible only in respective classes and is bound to the entire class hierarchy.

However, Hook methods are overridden in sub classes. Hence, their implementation in base class and subclasses can be independently modified without affecting the other implementations. Hence, MI is further split into Adapted Method Implementation (AMI) and Method Implementation (MI) in the proposed scheme. Therefore, implementation can be read or modified by lock modes (RAMI, RMI, MMI). There is only read mode available for adapted method implementations. MMI includes modification of hook methods and methods defined in this class.

However, when transactions request to modify adapted attributes in the respective classes where they are defined and try to read them in the class where they are adapted, should not be allowed to maintain consistency. This is applicable to modification of signatures and implementation of methods too to maintain consistency. Hence, the compatibility of lock modes on these attributes, signatures and implementations of these methods at both the classes where they are defined and where they are read, need to be checked to maintain consistency. Table 1 defines the compatibility matrix defined for classes. Table 2 defines the compatibility matrix for classes where the attributes and methods (signature and implementation) are defined against where they are adapted, where Y means two methods commute always and ∆ means that the two methods commute only when they access disjoint portions of an object. The hierarchy of granules in design time requests in Malta and Martinez [7], Jun [8] and proposed scheme can be summarized as given in fig 2.



(a) Malta & Martinez Scheme (b) Jun Scheme (c) Proposed Scheme

Fig. 2. Hierarchy of granules of design time requests

The semantics of the various modes defined are as follows:

1. **RAA** – Read Adapted Attributes – Read attributes defined in base class and component class adapted in to this class
2. **RA** – Read Attribute – Read attributes defined in this class.
3. **MA** – Modify Attribute – Modify attributes defined in this class.
4. **RAMS** – Read Adapted Method Signature – Read signature of template methods and hook methods adapted from other classes by inheritance and methods from component classes.
5. **RMS** – Read Method Signature – Read signature of methods defined in this class.

Table 1. Compatibility matrix among design time requests

Requested mode

Current mode

	RAA	RA	MA	RAMS	RMS	MMS	RAMI	RMI	MMI	RCR	MCR
RAA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Δ
RA	Y	Y	Δ	Y	Y	Y	Y	Y	Y	Y	Δ
MA	Y	Δ	Δ	Y	Y	Y	Y	Y	Y	Y	Δ
RAMS	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Δ
RMS	Y	Y	Y	Y	Y	Δ	Y	Y	Δ	Y	Δ
MMS	Y	Y	Y	Y	Δ	Δ	Y	Δ	Δ	Y	Δ
RAMI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Δ
RMI	Y	Y	Y	Y	Y	Δ	Y	Y	Δ	Y	Δ
MMI	Y	Y	Y	Y	Y	Y	Y	Δ	Δ	Y	Δ
RCR	Y	Y	Y	Y	Y	Δ	Δ	Y	Δ	Y	Δ
MCR	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ

Table 2. Compatibility matrix between defined class and adapted classes

Lock mode in adapted class

Lock mode in defined class

	RAA	RAMS	RAMI	RMI	MMI
MA	Δ	Y	Y	Y	Y
MMS	Y	Δ	Δ	Δ	Δ
MMI	Y	Y	Δ	Y	Y

6. **MMS**– Modify Method Signature – Modify signature of methods defined in this class.
7. **RAMI** – Read Adapted Method Implementation - Read implementation of template methods adapted from other classes by inheritance and methods from component classes.
8. **RMI** – Read Method Implementation - Read implementation of hook methods adapted from other classes by inheritance and methods defined in this class.
9. **MMI** – Modify Method Implementation - Modify implementation of hook methods adapted from other classes by inheritance and methods defined in this class.
10. **RCR** – Read class Relationship – Reading the definition of class, its relationship with other classes in the class lattice.
11. **MCR**– Modify Class Relationship - changing edges or changing nodes in class diagram.

The modification to class relationships can be modifying edges or modifying nodes as mentioned in section 2.1. The concurrency control is imposed by using Relationship Vectors (RV). The relationship vector of a class can be defined as a set of classes that are subclasses and/or composite classes which are directly or indirectly affected by any changes on this class. For e.g. in fig 3, classes E and H are sub classes of A by direct inheritance and indirect inheritance respectively. Then RV [A] includes A,E

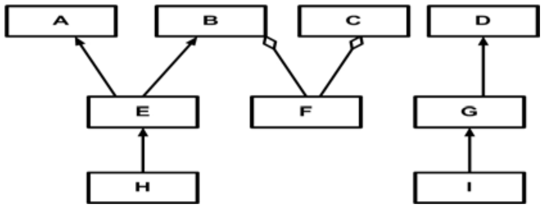


Fig. 3. Sample class diagram

and H as they are the classes which will be affected by any modification of class relationship done in A. A is also included to avoid parallel changes on A.

The relational vectors for the sample class diagram in fig 2 are given as follows:

[A] = [A, E, H]; RV [B] = [B, E, H]; RV [C] = [C, F]; RV [D] = [D, G, I];
RV [E] = [E, H]; RV [F] = [F]; RV [G] = [G.I]; RV [H] = [H]; RV [I] = I

From the list of relational vectors, it can be inferred that the dependency is more when the classes are in the top of the class lattice and hence more no of classes are affected when their class relationships are modified like A, B, D. In the lowest level of classes for e.g. H, F and I, only those classes are affected when their class relationships are modified. Then commutativity of Relational Vectors for the sample diagram can be constructed as follows:

Table 3. Example of Commutativity of relationship vector for sample class diagram (fig 3)

	A	B	C	D	E	F	G	H	I
A	N	Y	Y	Y	N	Y	Y	N	Y
B	Y	N	Y	Y	N	Y	Y	N	Y
C	Y	Y	N	Y	Y	N	Y	Y	Y
D	Y	Y	Y	N	Y	Y	N	Y	N
E	Y	Y	Y	Y	N	Y	Y	N	Y
F	Y	Y	Y	Y	Y	N	Y	Y	Y
G	Y	Y	Y	Y	Y	Y	N	Y	N
H	Y	Y	Y	Y	Y	Y	Y	N	Y
I	Y	Y	Y	Y	Y	Y	Y	Y	N

Let us consider the various scenarios to show how jun's scheme and proposed scheme works: Let T1 be a transaction arriving at t. Let T2 and T3 be transactions arriving at t+1. Let us assume that each transaction takes atleast 1 second to complete. Let *class name: [tran-name, lock type (item name)]* be the format for design time request. *Item name* refers to the name of the attribute or method which is accessed. Let us consider fig 1. Let the base class be C1.Let its sub class be C2. T1 requests C1. T2 and T3 requests C2. The scenarios will show how the proposed scheme improves consistency wherever necessary and improves concurrency using runtime information.

	Jun's Scheme	Proposed Scheme
1. t: C1:[T1,MS(M1)]	[T1,MS(M1)]	[T1,MS(M1)]
t+1:C2:[T2,RS(M1)]	[T1,MS(M1)] [T2,RS(M1)] // allowed	[T1,MS(M1)] //[T2,RS (M1)] is blocked as M1 is a template method and consistency is affected.
2 t: C1:[T1,MA(A1)]	[T1,MA(A1)]	[T1,MA(A1)]
t+1:C2:[T2,RA(M1)]	[T1,MA(A1)] [T2,RA(A1)] // allowed	[T1,MA(A1)] //[T2,RA (A1)] is blocked as A1 is an attribute and its consistency is affected.
3. t: C1:[T1,MI(M1)]	[T1,MI(M1)]	[T1,MI(M1)]
t+1:C2:[T2,RAMI(M1)]	[T1,MI(M1)] [T2,RAMI(M1)] // allowed	[T1,MI(M1)] //[T2,RAMI (M1)] is blocked as M1 is a template method and consistency is affected.
4. t: C1:[T1,MI(M2)]	[T1,MI(M2)]	[T1,MI(M2)]
t+1:C2:[T2,MI(M2)]	[T1,MI(M2)] [T2,MI(M2)] // allowed	[T1,MI(M2)][T2,MI(M2)] // allowed as M2 is a hook method and it can modify implementations in base class and subclass independently.
5. t: C2:[T2,MA(B1)]	[T2,MA(B1)]	[T2,MA(B1)]
t+1:C2:[T3,RAA(A1)]	[T2,MA(B1)] [T3,RAA(A1)] // allowed only if AAV is present	[T2,MS(B1)][T3,RAA(A1)] // allowed by using different lock modes
6. t: C2:[T2,MS(M3)]	[T2,MS(M3)]	[T2,MS(M3)]
t+1:C2:[T3,MI(M2)]	[T1,MS(M3)] [T3,MI(M2)] // allowed only if MAV is present.	[T1,MS(M3)] [T3,MI(M2)] // allowed by using different lock modes.
7. t: C2:[T2,RAMS(M1)]	[T2,RAMS(M1)]	[T2,RAMS(M1)]
t+1:C2:[T3,MS(M3)]	[T2,RAMS(M1)] [T3,MS(M2)] // allowed only if MAV is present.	[T2,RAMS(M1)] [T3,MS(M2)] // allowed by using different lock modes.
8. t: C2:[T2,RAMI(M2)]	[T2,RAMI(M2)]	[T2,RAMI(M2)]
t+1:C2:[T3,MI(M3)]	[T2,RAMI(M2)] [T3,MI(M3)] // allowed only if MAV is present.	[T2,RAMI(M2)] [T3,MI(M3)] // allowed by using different lock modes.

From fig2,

Let T1 tries to change the relationship R between A and E. T2 moves method M from G to I.

9. t: A:[T1,MCR(R)]	//not defined	[T1,MCR(R)]
t+1:G:[T2,MCR(M)]	//not defined	[T1,MCR(R)] [T2,MCR(M)] //allowed using RV
10. T: A:[T1,MCR(R)]	//not defined	[T1,MCR(R)]
t+1:E:[T2,MCR(M)]	//not defined	[T1,MCR(R)] [T2,MCR(M)] // not allowed using RV

Using the commutativity of lock modes, a finer granularity lock can be obtained. The lock granularity in the proposed work is one of MA, MMS, MMI, MCR and RAA, RA, RAMS, RMS, RAMI, RMI and RCR. Whenever a design time request is made, table 1 is checked for compatibility. If it is sub class or composite class and lock mode is one of RAA, RAMS, RMI and MMI or if it is base class or component class and lock mode is one of MA, MMS, MMI, table 2 is checked for compatibility.

3.2 Concurrency between Design Time Requests and Runtime Requests

The concurrency between instance access and design time access can be maximized using lock modes defined in table 4. The lock modes defined below and their compatibility is possible with the help of AAV, MAV and RV.

Table 4. Commutativity among design time requests and runtime requests

	RAA	RA	MA	RAMS	RMS	MMS	RAMI	RMI	MMI	RCR	MCR	I
RAA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Δ	Y
RA	Y	Y	Δ	Y	Y	Y	Y	Y	Y	Y	Δ	Y
MA	Y	Δ	Δ	Y	Y	Y	Y	Y	Y	Y	Δ	Δ
RAMS	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Δ	Y
RMS	Y	Y	Y	Y	Y	Δ	Y	Y	Δ	Y	Δ	Y
MMS	Y	Y	Y	Y	Δ	Δ	Y	Δ	Δ	Y	Δ	Δ
RAMI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Δ	Y
RMI	Y	Y	Y	Y	Y	Δ	Y	Y	Δ	Y	Δ	Y
MMI	Y	Y	Y	Y	Y	Y	Y	Δ	Δ	Y	Δ	Δ
RCR	Y	Y	Y	Y	Y	Δ	Δ	Y	Δ	Y	Δ	Y
MCR	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
I	Y	Y	Δ	Y	Y	Δ	Y	Y	Δ	Y	Δ	Δ

4 Correctness of the Algorithm

As the proposed scheme is based on standard two phase locking, the serializability is guaranteed [14]. Thus it is sufficient to show that all access conflicts are detected. Based on the description of the paper, it is clear that the lock modes in the proposed scheme detects all read- write and write –write conflicts among design time requests and instance accesses. The smallest granules that can be accessed during design time within a class are attributes and methods. The proposed scheme checks the compatibility against read - write conflicts and write-write conflicts of these granules by checking the compatibility matrix of the defined class, and the compatibility matrix of adapted class that could read or write the same granules. Thus, the overhead of using AAV and MAV are reduced by defining separate lock modes for attributes and methods in defined class and adapted class. This was overlooked in Jun [8] that could lead to inconsistency. Similarly access of class relationships in Jun [8] required locking of entire class diagram. However, in the proposed scheme locking of only dependent class hierarchy is enough, rather than the entire class diagram.

5 Conclusion

In this paper, a concurrency control scheme is proposed on multiple granularities to provide fine granularity among design time requests and instance accesses. The proposed scheme imposes concurrency control on the write-to-read conflicts between classes related by inheritance and composition. It minimizes the need for AAV and MAV used in jun's scheme by proposing rich set of lock modes based on semantics of object oriented concepts. Fine granularity on modifying class relationships is proposed by defining relation vectors. The objective of this paper is to provide finest granularity on all lock modes to provide highest concurrency, however with the overhead of maintaining AAV, MAV and RV. Future work is aimed at minimizing this overhead.

References

1. Gray, J.N., Lorie, R.A., Putzolu, G.R., Traiger, L.I.: Granularity of locks and degrees of consistency in shared database. In: Nijssen, G.M. (ed.) Modeling in Database management system, pp. 393–491. Elsevier, North Holland (1978)
2. Garza, F., Kim, W.: Transaction management in an object oriented database system. In: Proc. ACM SIGMOD Int'l Conference, Management Data (1987)
3. Kim, W., Bertino, E., Garza, J.F.: Composite Objects revisited. In: Object Oriented Programming, Systems, Languages and Applications, pp. 327–340 (1990)
4. Geetha, V., Sreenath, N.: Impact of Object Operations and Relationships in Concurrency Control in DOOS. In: Kant, K., Pemmaraju, S.V., Sivalingam, K.M., Wu, J. (eds.) ICDCN 2010. LNCS, vol. 5935, pp. 258–264. Springer, Heidelberg (2010)
5. Jun, W., Gruenwald, L.: An Effective Class Hierarchy Concurrency Control Technique in Object – Oriented Database Systems. Elsevier Journal of Information and Software Technology, 45–53 (1998)
6. Lee, S.Y., Liou, R.L.: A Multi-Granularity Locking model for concurrency control in Object – Oriented Database Systems. IEEE Transactions on Knowledge and Data Engineering 8(1) (1996)
7. Malta, C., Martinez, J.: Automating Fine Concurrency Control in Object Oriented Databases. In: 9th IEEE Conference on Data Engineering, Austria, pp. 253–60 (1993)
8. Jun, W.: A multi-granularity locking-based concurrency control in object oriented database system. Elsevier Journal of Systems and Software, 201–217 (2000)
9. Saha, D., Morrissey, J.: A self – Adjusting Multi-Granularity Locking Protocol for Object – Oriented Databases. IEEE, Los Alamitos (2009)
10. Riehle, D., Berczuk, S.P.: Types of Member Functions in C++, Report (2000)
11. Riehle, D., Berczuk, S.P.: Properties of Member Functions in C++, Report (2000)
12. Szyperski, C., Gruntz, D., Murer, S.: Component Software – Beyond object -oriented programming, 2nd edn. Pearson Education, London (2002)
13. Banerjee, J., Kim, W., Kim, H.J., Korth, H.F.: Semantics and Implementation of Schema evolution in Object–Oriented Databases. In: Proc. ACM SIGMOD Conference (1987)
14. Eswaran, K., Gray, J., Lorrie, R., Traiger, I.: The notion of consistency and predicate locks in a database system. ACM Communications 19(11), 624–633 (1976)

Contention-Free Many-to-Many Communication Scheduling for High Performance Clusters

Satyajit Banerjee*, Atish Datta Chowdhury*
Koushik Sinha[1],**, and Subhas Kumar Ghosh[2]

[1] Honeywell Technology Solutions, Bangalore, India
sinha_kou@yahoo.com
[2] Siemens Corporate Research and Technologies,
Bangalore, India
subhas.k.ghosh@gmail.com

Abstract. In the context of generating efficient, contention free schedules for inter-node communication through a switch fabric in cluster computing or data center type environments, *all-to-all* scheduling with *equal sized* data transfer requests has been studied in the literature [1, 3, 4]. In this paper, we propose a *communication scheduling module* (CSM) towards generating contention free communication schedules for many-to-many communication with arbitrary sized data. Towards this end, we propose three approximation algorithms - PST, LDT and SDT. From time to time, the CSM first generates a bipartite graph from the set of received requests, then determines which of these three algorithms gives the best approximation factor on this graph and finally executes that algorithm to generate a contention free schedule. Algorithm PST has a worst case run time of $O(\max(\Delta|E|, |E|\log(|E|)))$ and guarantees an approximation factor of $2H_{2\Delta-1}$, where $|E|$ is the number of edges in the bipartite graph, Δ is the maximum node degree of the bipartite graph and $H_{2\Delta-1}$ is the $(2\Delta-1)$-th harmonic number. LDT runs in $O(|E|^2)$ and has an approximation factor of $2(1+\tau)$, where τ is a constant defined as a *guard band* or *pause time* to eliminate the possibility of contention (in an apparently contention free schedule) caused by system jitter and synchronization inaccuracies between the nodes. SDT gives an approximation factor of $4\log(w_{max})$ and has a worst case run time of $O(\Delta|E|\log(w_{max}))$, where w_{max} represents the longest communication time in a set of received requests.

Keywords: Many-to-many scheduling, contention free schedule, switch scheduling, switch fabric, cluster computing, data centers, approximation algorithms.

1 Introduction

Recent years have seen cluster computing obtain tremendous impetus for usage ranging from the hosting of web services to general computing applications, scientific and other computation intensive applications, sensor data processing and video rendering,

* Contributed to this work while at Honeywell Technology Solutions, Bangalore, India. Email: satyajitb@gmail.com, adattachowdhury@yahoo.com
** Corresponding author.

to name a few. A pertinent question in designing a cluster is whether there is any need of a *communication scheduling module* (CSM) that can do a better job than the in-built logic of the switch fabric at generating a contention free communication schedule for a set of data transfer requests. Such a module would take as input the set of application communication requests and ideally generate a schedule with optimal latencies.

An alternative to the CSM could be to have the applications generate data exchange schedules [12, 14] that take into account the communication overheads. However, this approach is not frequently used due to the difficulties in solving an optimization problem that considers communication overheads.

Thus, the popular approach involves having the application generate a data redistribution that ignores the communication overheads and then rely completely on the underlying switched interconnection network using high performance switches [8, 11], for handling the communications. Unfortunately, even with the availability of high performance switches it is possible that due to the adverserial nature of data forwarding by some of the compute nodes, messages at the switch are highly contending. Thus, delegating the responsibility of communication scheduling to the underlying switch fabric may cause sub-optimal latencies, which in turn, may lead to failure to meet the application QoS requirements.

Even under an assumption of ideal *output queued* (OQ) switch with unlimited output buffers, the limited speed of the output links will introduce suboptimal latencies unless the communications are properly scheduled. An example of this that has been researched is the regular *all-to-all* communication on switched interconnects [1, 3, 4]. However, for many cluster applications, an all-to-all communication pattern may not exist. Instead, their communication patterns can be better described as *many-to-many* communication with arbitrary sized data. Examples of such communication patterns may be found in dynamic load balancing through data/task migration, replication and dynamic redistribution of *virtual machines* (VMs) in large data centers and mesh based numerical simulations/computations [12, 14–16].

1.1 Our Contribution

In this paper, we focus on designing contention free schedules. We propose a *communication scheduling module* (CSM) that is capable of generating contention free communication schedules for many-to-many communication with arbitrary sized data. Towards this end, we propose three approximation algorithms - PST, LDT and SDT. The CSM takes as input a set of communication requests, determines the algorithm that gives the best approximation factor for the given set of requests and then executes that algorithm to generate a contention free schedule. The CSM generates a schedule under the following two assumptions: i) the contention free schedule generated by the CSM is executed by the underlying switch fabric without any modification and, ii) the schedule is executed by the switch fabric immediately with no additional delays (e.g., delays resulting from buffering of the data to optimize messaging overhead). A survey of the literature shows that these are reasonable assumptions to make for most current IQ, OQ and CIOQ switches [4, 8, 11].

A significant difference between existing scheduling algorithms for IQ switches and our problem is as follows: in IQ switching the exact amount of data to be transferred is unknown except for some bound on the admissible rate of data arrival on the ports. However, in our problem the amount of data to be transferred between every pair of ports is given *a priori*. With this observation, instead of randomized edge coloring based techniques [5], we can use deterministic maximal matching based algorithms to generate optimal schedules. In this paper, we use maximal matching algorithms instead of costly maximum weight matching (MWM) algorithms [5].

We model the many-to-many communication requirements in the form of a bipartite graph, and generate communication schedules within a certain approximation factor of the optimal schedule (having minimum total communication latency). Algorithm *Partial Shuffle Transfer* (PST) guarantees an approximation factor of $2H_{2\Delta-1}$, where Δ is the maximum node degree of the bipartite graph and $H_{2\Delta-1}$ is the $(2\Delta-1)$-th harmonic number. Algorithm PST has a worst case run time of $O(\max(\Delta|E|, |E|\log(|E|)))$, where $|E|$ is the number of edges in the bipartite graph. PST is suitable for application scenarios where it is unlikely to have a bipartite graph with high value of Δ. Such requirements may arise, e.g., in the context of redistribution of VMs in physical machines for dynamic load balancing where the objective is to perform incremental and regular correction rather than complete re-shuffling. Algorithm *Large Data Transfer* (LDT) runs in $O(|E|^2)$ and has an approximation factor of $2(1+\tau)$, where τ is a *pause time* constant to eliminate the possibility of contention (in an apparently contention free schedule) caused by system jitter and synchronization inaccuracies between the nodes. This may be useful for scenarios where there is a need for inter-node migration of large volumes of data from time to time (e.g., data/process replication) and/or where the compute nodes are reasonably well synchronized by some time synchronization protocol. Algorithm *Small Data Transfer* (SDT) has been designed for scenarios where the amount of time required to complete each of the communication requests is small (e.g., in some mesh based scientific computations). SDT has a worst case run time of $O(\Delta|E|\log(w_{max}))$ and an approximation factor of $4\log(w_{max})$, where w_{max} represents the largest transfer time in the set of received communication requests.

2 System Model

We consider a cluster of compute nodes linked through a fully duplex, switched interconnection network (typically forming a LAN) so as to form a connected graph. Each switch has m number of up-link ports and one down-link port. We assume that the switches are arranged hierarchically [3, 4, 11] to form a complete m-ary tree. Compute nodes are connected to the up-link ports of leaf level switches and the *communication scheduling module* (CSM) runs on a compute node/device attached to the down-link port of the *root switch*. We assume that the compute nodes connected to the leaf level switches are synchronized with the possibility of some clock drift (e.g., through the use of the *network time protocol*).

We make the following assumptions on the *configuration* of the hierarchically organized switches:

1. The down-link speed of a switch is m times its up-link speed.
2. The up-link speed of a switch at depth/level i is same as the down-link speed of a switch at depth $(i+1)$.

The above assumptions related to the switch fabric configuration ensure congestion free data transfer from the leaf level switches to the *root switch* and vice versa. Fig. 1 depicts a two level hierarchical switched interconnection network with $m = 4$. The speed of each up-link of the leaf level switches is denoted by x. Accordingly, the corresponding down-link speed of the each of the leaf-level switches is $4x$. The up-link speeds of the root switch are $4x$ and its down-link speed is thus $16x$. The CSM is connected to the root switch through this $16x$ speed link.

We also make the following two assumptions on the behavior of the *in-built logic* of the *switch fabric*. Typically, these assumptions hold good for almost all the existing switch variants e.g IQ, OQ and CIOQ [8, 11].

1. The switch fabric executes any contention free data exchange schedule between its up-link ports without any modification.
2. The switch fabric does not introduce any additional delay (e.g., through buffering of data until a minimum message size is reached) while executing a contention free data exchange between its up-link ports.

The above assumptions related to the switch fabric behavior ensure that the individual switches do not introduce any additional communication overhead in a contention free data exchange schedule that is presented to the fabric.

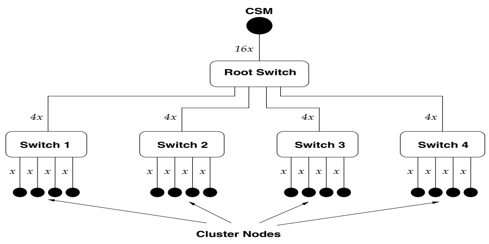


Fig. 1. A 2-level hierarchical switched interconnection tree with $m = 4$

Note that the above assumptions related to the configuration and behavior of the switch fabric allow us to abstract the switch fabric as a *single switch* which gracefully executes the contention free data exchange computed by the CSM without any modification and also without incurring any additional delay. The procedure for generating a contention free communication schedule using the CSM is as follows:

– Whenever a compute node has a need for data communication, it sends information about its communication requests up the interconnection tree to the *root switch* in the form of (source node, destination node, communication time) tuples.
– Next, the CSM attached to the root switch collects all the tuples and runs an appropriate algorithm to arrive at some optimal contention free schedule assuming the slowest link speed for each communication request. For this, it computes the time required for each data transfer request based on the slowest link speed in the path from the sender to the receiver (e.g., in Fig. 1 the slowest link speed is x for a path between any pair of nodes). The schedule computation is done either when

a sufficient number of requests have been received or after a time out period since the first received request. The schedule is then returned to the *root switch*. The *root switch* then sends this schedule down the tree to the requesting compute nodes.
- The compute nodes upon receiving the schedule from the root switch initiate their data transmissions according to the schedule.
- The above steps are repeated by the compute nodes and the root switch whenever there is a new communication request generated at a compute node.

3 Problem Formulation

In a cluster, if a node i needs to send to node j some data that takes t amount of time to *transmit* with the available network speed, then we denote such a communication request as (i, j, t). Now, given a set of communication requests received by the CSM, our goal is to arrive at a transmission schedule that minimizes the total time required to migrate all the data in that set. As contention adds on to the incurred communication latencies, we focus on designing contention free schedules. In this context, the use of *matching* based approaches is particularly attractive.

We associate with each node i, two logical ports - an *out-port* i_{out} and an *in-port* i_{in}. We can then view each request of the form (i, j, t) as a data transfer from out-port i_{out} to in-port j_{in} for duration t. We can thus represent such a set of port-level communication requests by a weighted bipartite graph $G = (O \cup I, E, w)$ where O and I are the sets of out-ports and in-ports respectively, E is the set of edges and w is a mapping defined as follows: for each port-level request (i_{out}, j_{in}, t) there is an edge $e_{ij} \in E$ having weight $w(e_{ij}) = t$. For rest of the paper, we adopt the following convention: i) we shall refer to the ports in O and I as *vertices* and the compute nodes as *nodes*, ii) we shall use the terms w_{ij} and $w(e_{ij})$ interchangeably. For such a bipartite graph, the idea is to generate a number (say L) of sets $M_k \subseteq E$, $1 \leq k \leq L$ such that:

1. $M_1, \ldots, M_L$ decompose an edge e_{ij} into L edges as $e_{ij}^{(1)}$, $e_{ij}^{(2)}, \ldots, e_{ij}^{(L)}$, with weight $w_{ij}^{(k)}$ assigned to edge $e_{ij}^{(k)}$ such that: $\sum\limits_{1 \leq k \leq L} w_{ij}^{(k)} = w_{ij}, w_{ij}^{(k)} \geq 0$.
2. Edges of non-zero weight in each M_k form a matching.

The output is a schedule $S = \{(M_1, T_1), (M_2, T_2), \ldots, (M_L, T_L)\}$ where $T_k = \max\{w_{ij}^{(k)}\}$ and $1 \leq k \leq L$, for executing a set of communication requests received by the CSM. The values $T_1, T_2, \ldots, T_L$ indicate that data needs to be transmitted for a maximum of T_k amount of time along each edge $e_{ij} \in M_k$ in the k^{th} step of the schedule. Since each M_k is a matching, the schedule S ensures contention free transmission of data amongst *perfectly synchronized* nodes. Thus the total communication time of data is the sum of all the T_k's.

Fig. 2 depicts a bipartite graph G representing the input port-level data communication requests and a possible schedule S consisting of a sequence of matchings M_1, M_2 and M_3 derived from G. Matching M_2 is executed after M_1 followed by M_3. In Fig. 2, for each matching only edges with non-zero weights have been shown. From Fig. 2 we see that the edge e_{43} is present in M_1 and M_2 with weights $w_{43}^{(1)}$ and $w_{43}^{(2)} = w_{43} - w_{43}^{(1)}$ respectively.

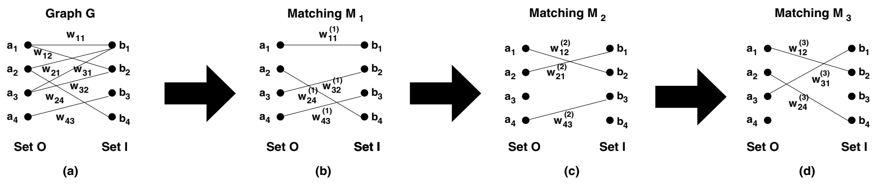


Fig. 2. A representative bipartite communication graph G and a possible schedule S derived from G

Unfortunately, while the resultant schedule is theoretically contention free, in practice, collisions may still occur due to system related issues associated with (i) clock drift and (ii) operating system jitter [3, 4, 10]. For tightly coupled systems or solutions with dedicated hardware for clock synchronization, synchronization overhead due to clock drift might be insignificant. However, that typically leads to costly solutions. Thus, *network time protocol* (NTP) [9] based solutions that in general provide relatively coarser granularity synchronization between the nodes are more popular and for the rest of the paper we assume the nodes to be using NTP. Hence, to ensure contention free transmissions, it is important to take into account the effect of system jitter and *partial synchronization* between the nodes arising from the use of NTP. Therefore, we introduce a *pause time* or *guard band* (τ) between the execution of two consecutive matchings M_i and M_{i+1} in S. We assume τ to be a constant. The value of τ depends on i) the maximum drift of the clocks between two consecutive invocations of NTP and, ii) system jitter. For the rest of the paper, we use *pause time* and *guard band* interchangeably.

A collision-free data migration amongst partially synchronized nodes can be guaranteed using the following rule: In each step i, all the vertices having some data to be sent through the edges of M_i start sending data at their local time $t_0 + \sum_{0<j<i} T_j + (i-1)\tau$ for duration at most T_i, where t_0 is the execution start time of schedule S. It is easy to see that the number of guard bands that need to be inserted in S is one less than the number of matching in the schedule. For example, in Fig. 2, the total transmission time corresponding to the schedule $S = \{M_1,\ M_2,\ M_3\}$ is:

$$\mathcal{T} = \max\,(w_{11}^{(1)}, w_{24}^{(1)}, w_{32}^{(1)}, w_{43}^{(1)}) + \max\,(w_{12}^{(2)}, w_{21}^{(2)}, w_{43}^{(2)}) + \max\,(w_{12}^{(3)}, w_{31}^{(3)}, w_{24}^{(3)}) + 2\tau,$$

where $\tau > 0$ is a constant denoting the pause time or duration of each guard band. Note that, for perfectly synchronized nodes $\tau = 0$. Therefore to minimize the total communication time we need to find a schedule S that minimizes both the total transfer time and the total pause time. That is, we can formulate our problem of finding an efficient communication schedule for data transfer in high performance clusters as:

$$\textit{minimize}\ \mathcal{T} = \sum_{1\le i\le L} T_i + \tau(L-1) \qquad (1)$$

4 Proposed Solution

For the ease of analysis, we assume that all w_{ij}'s are integers. If w_{ij}'s are not integers then we first scale the values with respect to the minimum value among the w_{ij}'s and then round up to get integer edge weights. Given such integral values of w_{ij}'s in G, it can be shown that the first component ($\sum T_i$) in equation 1 can be optimized by considering edge coloring of the corresponding bipartite multigraph of G [6, 7]. On the other hand, optimizing the second component $\tau(L-1)$ is equivalent to minimizing the number of matching sets that are internally contention free. It can thus be separately optimized by considering edge coloring of the bipartite simple graph G.

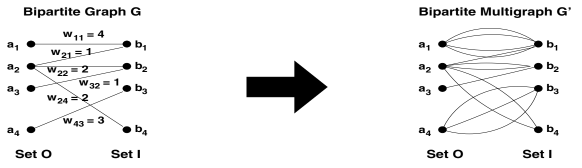


Fig. 3. A weighted bipartite graph and its equivalent bipartite multigraph

Fig. 3 shows a weighted bipartite graph $G = (O \cup I, E, w)$ with integer edge weights and the corresponding multigraph $G' = (O \cup I, E')$ derived from G. For each edge $e_{ij} \in E$ of weight w_{ij}, G' has w_{ij} edges between vertices i and j. Thus, if the total edge weight incident on a vertex i in G is λ_i, then the vertex degree of i in G' is λ_i. For example, in Fig. 3 the sum of edge weights incident on vertex b_1 is 5 (since $w_{11} = 4$ and $w_{21} = 1$) and hence in G' the degree of vertex b_1 is 5. Similarly, the vertex degree of a_2 in G' is 5 (since $w_{21} = 1, w_{22} = 2$ and $w_{24} = 1$). We denote the maximum vertex degree of G by Δ and that of G' by Δ_w where $\Delta_w = \max_i(\lambda_i)$. Note that because of integer edge weights in G, $\Delta_w \geq \Delta$. For example, $\Delta = 3$ and $\Delta_w = 5$ in Fig. 3.

The fastest known exact algorithm for edge coloring bipartite graphs runs in time $O(E \log(\Delta))$ [6]. Although, this serves our purpose for the second component, its complexity becomes unacceptable (super polynomial time) for the first component when edge multiplicities are encoded as binary numbers in the input. The best known approximation algorithm for the first component runs in $O(|V| \log(w_{max}))$ time [7]. One might argue that the technique of storing parallel edges as described in [7] can be used in [6] to design an exact algorithm with run time polynomial in $|V|$ and $\log(w_{max})$. But, this is non-trivial and on top of that our problem is at least as hard the first component alone, if not harder. Thus, we restrict ourselves only to approximation algorithms for the overall optimization problem denoted by equation 1.

4.1 CSM Module Description

In this paper we propose three different approximation algorithms for handling various types of communication requests determined by the combination of i) the properties of graph $G = (V, E, w)$ and, ii) the pause time τ.

Fig. 4 depicts a block diagram of the CSM. The CSM takes as input i) the set of communication requests generated by the application as a bipartite graph $G = (V, E, w)$ and, ii) the pause time τ. This input is analyzed by the *algorithm choice unit* to determine which of our three proposed algorithms (PST, LDT and SDT described later in this section) would give the best approximation bound. To determine this, it first evaluates Δ and w_{max}. It then computes the approximation bound for each of the three

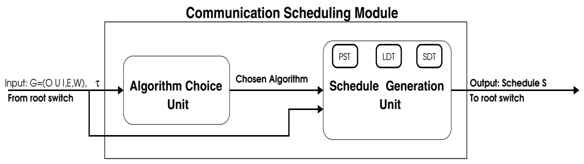


Fig. 4. Block diagram of our proposed CSM module

algorithms. If there is a tie, then the algorithm with the lowest worst-case runtime is chosen. Together with the original input graph and τ, this choice of the algorithm then serves as the input to the *schedule generation unit*. The output of the *schedule generation unit* is the schedule S which is returned to the root switch for dissemination among the compute nodes.

4.2 Lower Bound on the Total Migration Time

We capture a lower bound on the total migration time of a schedule that follows from the structure of the bipartite graph induced by the set of data transfer requests.

Lemma 1. *No optimal schedule can guarantee a total migration time less than* $\Delta_w + \tau(\Delta - 1)$.

Proof. For any (optimal) schedule, the number of matchings and transfer time are respectively at least Δ and Δ_w, both of which follow from Konig's theorem on edge coloring bipartite graphs [13] when applied on graphs G and G' respectively. Thus, for any schedule, the migration time is at least $\Delta_w + \tau(\Delta - 1)$. □

We use the above result while calculating the approximation factor of the algorithms presented in the subsequent sections.

4.3 Algorithm Partial Shuffle Transfer (PST)

For certain envisioned application scenarios such as redistributing VMs in physical machines [16], it is unlikely to have an input graph with high value of Δ as dynamic load balancing algorithms are meant for incremental correction and not complete re-shuffling [12, 14]. Our first algorithm called *Partial Shuffle Transfer* (PST) is designed for such applications. Algorithm PST ensures a good approximation guarantee when Δ is small.

```
1 Initialize, schedule $S \leftarrow \emptyset$; $i \leftarrow 1$
2 Sort the edges in the descending order of their weights
3 while there exists some edge yet to be chosen do
4     Try choosing edges in the sorted order until a maximal matching $M_i$ is formed
5     Let $w_{max}^{(i)} = \max_{e_{kl} \in M_i} (w_{kl})$      ▷ maximum edge weight among the edges in $M_i$
6     Append the tuple $(M_i, w_{max}^{(i)})$ to the schedule $S$; $i \leftarrow i + 1$
7 end
8 Output the resultant schedule $S$
```

Algorithm 1. Algorithm PST

4.4 Analysis of Algorithm PST

Lemma 2. *Algorithm* PST *runs in* $O(\max(\Delta|E|, |E|\log(|E|)))$ *time.*

Proof. Sorting takes $O(|E|\log(|E|))$ steps. And, in a phase, a vertex abstains from contributing an edge in the maximal matching only if all its neighbors contribute. Thus a vertex does not participate in forming the maximal matching at most for $\Delta - 1$ phases and in all other phases it has to necessarily participate. Therefore, maximum number of phases is $2\Delta - 1$. Now finding a maximal matching in each phase takes $O(|E|)$ and thus the running time of the algorithm (without considering the time required for sorting the edges in the first step) is $O(\Delta|E|)$. Hence the total running time of the Algorithm PST is $O(\max(\Delta|E|, |E|\log(|E|)))$. □

Lemma 3. *Algorithm* PST *guarantees* $2H_{2\Delta-1}$ *approximation factor.*

Proof. Let M_i be the maximal matching chosen in the i^{th} phase and $e_i^* \in M_i$ be of maximum weight t_i^* (which is also equal to T_i by definition). Since, e_i^* was not chosen in the previous phases, some edge from each of the previous phases, having weight at least t_i^* had a common vertex with e_i^*. Therefore, the total weight of the edges incident on one of the end vertices of e_i^* is at least $it_i^*/2$. This implies $2\Delta_w \geq it_i^*$ and hence $T_i \leq 2\Delta_w/i$. Using Lemma-2, the number of maximal matchings in the schedule is at most $2\Delta - 1$ and from the above argument transfer time $\sum_i T_i$ is at most $2\Delta_w H_{2\Delta-1}$, where $H_{2\Delta-1}$ is the $(2\Delta - 1)$-th *harmonic number*. Thus, the total migration time is less than equal to $(2\Delta_w H_{2\Delta-1}) + \tau(2\Delta - 2)$. Hence, the approximation factor is $\frac{(2\Delta_w H_{2\Delta-1})+\tau(2\Delta-2)}{\Delta_w+\tau(\Delta-1)} \leq 2H_{2\Delta-1}$. □

4.5 Algorithm Large Data Transfer (LDT)

In certain applications, there is a need for inter-vertex migration of large volumes of data from time to time (e.g., data/process replication). Also, it may so happen that in some application scenarios the nodes are reasonably well synchronized by some time synchronization protocol. In such situations, the pause time is typically order magnitude smaller than the individual edge weights of the input graph G. For such application scenarios, we present below algorithm *Large Data Transfer* (LDT) to generate contention free data exchange schedules which may have a better approximation factor than algorithm PST.

4.6 Analysis of Algorithm LDT

Lemma 4. *Algorithm* LDT *runs in* $O(|E|^2)$ *time.*

Proof. Finding a maximal matching in each step takes $O(|E|)$ time and in each step at least one edge having the minimum weight in the chosen matching is fully exhausted. Hence the worst case running time of the algorithm is $O(|E|^2)$. □

Lemma 5. *Algorithm* LDT *guarantees* $2(1 + \tau)$ *approximation factor.*

```
1  Initialize, schedule $S \leftarrow \emptyset$; $i \leftarrow 1$
2  while there exists an edge $e_{kl} \in E$ with $w_{kl} > 0$ do
3      Find a maximal matching $M_i$ from the set of edges $e_{kl} \in E$ with $w_{kl} > 0$
4      Let $w^{(i)}_{min} = \min_{e_{kl} \in E} (w_{kl})$        ▷ minimum edge weight of the edges in $M_i$
5      Append the tuple $(M_i, w^{(i)}_{min})$ to the schedule $S$
6      foreach $e_{kl} \in M_i$ do
7          $w_{kl} \leftarrow w_{kl} - w^{(i)}_{min}$
8      end
9      $i \leftarrow i + 1$
10 end
11 Output the resultant schedule $S$
```

Algorithm 2. Algorithm LDT

Proof. Consider the equivalent multigraph G' derived from G. Let Δ_w be the maximum vertex degree of G'. Then, a vertex v that has degree Δ_w in G' does not participate for at most $\Delta_w - 1$ communication steps and participates for at most Δ_w communication steps. Therefore, the number of matchings in the schedule can be at most $2\Delta_w - 1$. Since, each matching is for unit time duration, total communication time is also $2\Delta_w - 1$. Hence the approximation factor is $\frac{(2\Delta_w - 1) + \tau(2\Delta_w - 2)}{\Delta_w + \tau(\Delta - 1)} \leq 2(1 + \tau)$. $\square$

4.7 Algorithm Small Data Transfer (SDT)

For applications where the quantum of time required to complete each of the communication requests is small (e.g., in mesh based numerical simulations [12, 15]), i.e., the maximum edge weight $w_{\max}$ of the input bipartite graph is small, the following algorithm may be a suitable choice. We call this algorithm as *Small Data Transfer* (SDT). The description of Algorithm SDT is given below.

Let $G_i = (I \cup O, E_i, w)$ be the subgraph of G where $E_i \subseteq E$ and for each $e \in E_i$, $2^{i-1} < w(e) \leq 2^i$. Thus, there can be a maximum of $\log(w_{max})$ such component graphs in G. Assume that G can be partitioned into k such component graphs $G_1, G_2 \ldots, G_k$, $1 \leq k \leq \log(w_{max})$. Let G'_i be the equivalent multigraph of G_i. We define $\Delta^{(i)}$ and $\Delta^{(i)}_w$) as the maximum vertex degree of G_i and G'_i respectively. It clearly follows from the definition of G_i that: i) $\Delta^{(i)} \leq \Delta$, ii) $\Delta^{(i)}_w \leq \Delta_w$, iii) $2^{i-1}\Delta^{(i)} < \Delta^{(i)}_w \leq 2^i\Delta^{(i)} < 2\Delta^{(i)}_w$ and, iv) $|E_i| \leq |E|$.

```
1  Initialize, schedule $S \leftarrow \emptyset$; $i \leftarrow 1$
2  Let $G$ consist of the $k$ component graphs $\{G_1, \ldots, G_k\}$        ▷ $1 \leq k \leq \log(w_{max})$
3  for $j = 1$ to $k$ do
4      while there exists some edge in $G_j$ yet to be chosen do
5          Choose a maximal matching $M_i$ from $G_j$
6          Let $w^{(i)}_{max} = \max_{e_{kl} \in M_i} (w_{kl})$        ▷ maximum edge weight among the edges in $M_i$
7          Append the tuple $(M_i, w^{(i)}_{max})$ to the schedule $S$
8          Remove $M_i$ from the edge set of $G_j$; $i \leftarrow i + 1$
9      end
10 end
11 Output the resultant schedule $S$
```

Algorithm 3. Algorithm SDT

4.8 Analysis of Algorithm SDT

Lemma 6. *Algorithm* SDT *runs in* $O(\Delta|E|\log(w_{\max}))$ *time.*

Proof. Partitioning G into its component graphs takes $O(|E|)$-time. For each component graph G_i, finding maximal matching in each step takes $|E_i|$ steps and number of steps can be at most $(2\Delta_i - 1)$ (refer to arguments in Lemma-2). Therefore, the time required to run the algorithm for all the components is at most $O(\Delta|E|\log(w_{\max}))$ (since $\Delta^{(i)} \leq \Delta$ and $|E_i| \leq |E|$). □

Lemma 7. *Algorithm* SDT *guarantees* $4\log(w_{\max})$ *approximation factor.*

Proof. The contribution of G_i in the resultant schedule in terms of number of matching and communication time is at most $2\Delta^{(i)} - 1$ and $2^i(2\Delta^{(i)} - 1)$ respectively. Note that, $2\Delta^{(i)} - 1 < 2\Delta - 1$ (since $\Delta^{(i)} \leq \Delta$) and $2^i(2\Delta^{(i)} - 1) < 4\Delta_w$ (since $2^i\Delta^{(i)} < 2\Delta_w^{(i)}$ and $\Delta_w^{(i)} \leq \Delta_w$). Therefore in the complete schedule, total number of matchings and the total communication time are at most $(2\Delta - 1)\log(w_{\max})$ and $4\Delta_w\log(w_{\max})$ respectively. Hence, the approximation factor is $\frac{4\Delta_w\log(w_{\max})+\tau((2\Delta-1)\log(w_{\max})-1)}{\Delta_w+\tau(\Delta-1)} \leq 4\log(w_{\max})$. □

5 Discussion on Hierarchical Schedule Generation

We have so far assumed a system model where the schedule generation is done at the CSM attached to the down-link of the root switch. However, it is also possible to envision a hierarchical computation model under the assumption that every switch in the switch fabric has either a CSM unit attached to it or contains the CSM as part of the in-built scheduling logic. A possible procedure to compute a globally contention free schedule in a m-ary switch tree may then be as follows:

- Whenever a compute node has some data communication requirements, it sends information about its communication requests to its parent switch.
- The CSM attached to the parent switch collects all the tuples from its children. If there exist some tuples that involve source-destination pairs within its children nodes, it computes a contention free communication schedule S' for those nodes using the appropriate choice of algorithm and returns S' to the appropriate children. If some destination nodes are not within its set of children, then the switch sends such tuples and the schedule S' computed by it to its parent switch.
- A parent switch receives all partial schedules S' and unscheduled tuples from its children and repeats the above process of partial schedule generation with the start time of its generated schedule offset by an additional τ amount from the maximum of the completion times of the received partial schedules. For those tuples with destination nodes outside the nodes in its subtree, the switch passes on such tuples along with all the partial schedules generated by it and the switches in its subtree, to its parent switch. The process is repeated until all tuples are scheduled. At every level of the switch tree, any partial schedule generated by a switch is passed down to the compute nodes in its subtree which immediately initiate data transmissions according to the received schedule(s).

The advantage of the above scheme is two-fold: i) the schedule generation can be distributed across the switch fabric and, ii) a pair of vertices can possibly begin their communication earlier if their request can be scheduled by a switch at a level greater than that of the root switch of the m-ary switch tree.

6 Conclusion

We have presented a solution to the problem of many-to-many communication in cluster computing environments with arbitrary sized data. Our proposed approach utilizes a schedule generation unit called the CSM to generate contention free schedules for communication requests received from time to time from the compute nodes of the cluster. The CSM unit consists of three scheduling algorithms called PST, LDT and SDT. Depending on the value of τ and values of Δ and the longest communication time w_{max} in the bipartite graph derived from a set of received requests, CSM first determines the algorithm that gives the best approximation bound. It then uses that algorithm to generate an efficient contention free schedule for a received set of communication requests.

References

1. Faraz, A., Patarasuk, P., Yuan, X.: Bandwidth efficient all-to-all broadcast on switched clusters. Intl. Journal of Parallel Programming 36(4), 426–453 (2008)
2. Fan, X., Jonsson, M., Hoang, H.: Efficient many-to-many real-time communication using an intelligent Ethernet switch. In: Proc. ISPAN, pp. 280–287 (2004)
3. Tam, A.T.-C., Wang, C.-L.: Contention-aware communication schedule for high-speed communication. Cluster Computing 6(4), 339–353 (2003)
4. Yang, Y., Wang, J.: Optimal all-to-all personalized exchange in self-routable multistage networks. IEEE Trans. on Parallel and Dist. Systems 11(3), 261–274 (2000)
5. Aggarwal, G., Motwani, R., Shah, D., Zhu, A.: Switch scheduling via randomized edge coloring. In: Proc. IEEE Symp. on Foundations of Comp. Science (FOCS), p. 502 (2003)
6. Cole, K.O.R., Schirra, S.: Edge-coloring bipartite multigraphs in $O(E \log D)$ time. Combinatorica 21, 5–12 (2001)
7. Sanders, P., Steurer, D.: An asymptotic approximation scheme for multigraph edge coloring. ACM Trans. on Algorithms 4(2), 21:1–21:24 (2008)
8. Firoozshahian, A., Manshadi, V., Goel, A., Prabhakar, B.: Efficient, fully local algorithms for CIOQ switches. In: Proc. INFOCOM, pp. 2491–2495 (2007)
9. NTP: The Network Time Protocol, http://www.ntp.org/
10. Operating System Jitter (2010), http://domino.research.ibm.com/comm/research_projects.nsf/pages/osjitter.Identifying.html
11. Cisco 12000 Series Gigabit Switch Routers (2010), http://www.cisco.com/warp/public/cc/pd/rt/12000/prodlit/gsr_ov.pdf
12. Touheed, N., et al.: A comparison of dynamic load-balancing algorithms for a parallel adaptive flow solver. Parallel Computing 26(12), 1535–1554 (2000)
13. Diestel, R.: Graph Theory, 4th edn. Springer, Heidelberg (July 2010)
14. Sinha, K., Datta Chowdhury, A., Banerjee, S., Ghosh, S.K.: Efficient load balancing on a cluster for large scale online video surveillance. In: Garg, V., Wattenhofer, R., Kothapalli, K. (eds.) ICDCN 2009. LNCS, vol. 5408, pp. 450–455. Springer, Heidelberg (2008)
15. Gupta, A., Luksch, P., Schmidt, A.C.: MethWerk: scalable mesh based simulation on cluster of SMPs. In: Proc. High Perf. Comp. in Science and Engineering, pp. 141–151 (2005)
16. Wilcox Jr., T.C.: Dynamic load balancing of virtual machines hosted on Xen, MS Thesis, Dept. of Computer Science, Brigham Young University (April 2009)

Recursive Competitive Equilibrium Approach for Dynamic Load Balancing a Distributed System

K. Shahu Chatrapati[1], J. Ujwala Rekha[2], and A. Vinaya Babu[2]

[1] Dept. of Computer Science and Engineering, JNTUH College of Engineering, Jagitial
[2] Dept. of Computer Science and Engineering, JNTUH College of Engineering, Hyderabad
shahujntu@gmail.com, ujwala_rekha@gmail.com, dravinayababu@gmail.com

Abstract. Load balancing is very important and a common problem in distributed systems. The load balancing mechanism aims to fairly distribute load across the resources so as to optimize a given objective function. The objective can be system optimality which tries to minimize mean response time of all users or individual optimality which tries to minimize each user's individual response time. The load balancing can be achieved either statically or dynamically. In this paper we review, competitive equilibrium (CE) approach for static load balancing and then propose recursive competitive equilibrium (RCE) approach for dynamic load balancing. A computer model is run to evaluate the performance of proposed RCE scheme with static scheme using Nash equilibrium (NE) approach and the static scheme using CE. The results show that static scheme using CE and dynamic scheme using RCE achieved both system optimality and individual optimality simultaneously, while NE scheme achieved only individual optimality. Moreover the performance of RCE scheme is higher than CE when communication overhead is less and almost same as CE scheme when the communication overhead is more.

Keywords: Distributed System, Load Balancing, Competitive Equilibrium, Recursive Competitive Equilibrium.

1 Introduction

The purpose of load balancing in a distributed system is to minimize the response time of a single application running in parallel on multiple computers. Load balancing can be achieved either statically or dynamically. In static load balancing, the load assigned to a computer is proportional to its processing capacity and remains the same throughout the duration of the application or job. Whereas, in dynamic load balancing the distribution of load is assessed periodically, and then adjusted such that the resulting distribution results in the reduction of the remaining execution time of the job.

Static load balancing is simpler and less overhead intensive to implement, than dynamic load balancing. However, often it is not possible to predict the runtime behavior of either the running application or the computational environment and can thus invalidate the assumptions under which static load distribution was made. Hence, subsequent to initial static load distribution, sometimes load imbalances develop, which might result in a less efficient system. Therefore, run-time monitoring and redistribution of workload among the processors can improve the efficiency of the system. However, any benefit that can be derived due to the redistribution of load needs to be weighed against the overhead associated with monitoring and redistribution.

From the optimization perspective, the objective of load balancing can be to provide a system-optimal solution, where the mean response time of all jobs is minimized, or an individual optimal solution where each job minimizes its own response time. There are studies on static load balancing that provide system-optimal solution ([2],[16]). In such schemes, some jobs may experience very longer response time than others, which may not be acceptable in current distributed systems. Few studies exist on static load balancing that provide individual optimal solution based on game-theoretic solutions ([1],[13],[15]). However, such schemes may not achieve system optimal efficiency. Competitive equilibrium approach for achieving both system optimal efficiency and individual optimality is proposed in ([7],[8]). However, it does not take into account run-time behavior.

Many dynamic load balancing schemes exist that either achieve system optimality ([9],[17]) or individual optimality ([3],[14]) but not both simultaneously. In this study, we propose recursive competitive equilibrium approach for dynamic load balancing that simultaneously achieves both system optimal efficiency and individual optimality.

This paper is organized as follows: In sections 2 and 3, we discuss competitive equilibrium theory, and recursive competitive equilibrium theory respectively. Section 4 presents the distributed system model considered in our study and formulates the objective function. Sections 5 and 6 describe how competitive equilibrium theory and recursive competitive equilibrium theory can be applied to static and dynamic load balancing problems respectively. Section 7 shows the performance study of Nash Equilibrium Scheme (NES), Competitive Equilibrium Scheme (CES), and Recursive Competitive Equilibrium Scheme (RCES). And finally the paper concludes with section 8.

2 Competitive Equilibrium Theory

Competitive equilibrium theory is a branch of mathematical economics to model prices for a whole economy. The theory dates back to 1870s and the credit for initiating the study can be attributed to French Economist Lêon Walras [10]. In Walrasian model the market consists of $\mathbf{m}$ agents and $\mathbf{n}$ divisible goods. Let $\mathbf{w}_{ij}$ and $\mathbf{x}_{ij}$ denote respectively, the non-negative endowment and consumption by agent *i*, relative to the good *j*. Let $\mathbf{p}_j$ denote the non-negative price associated to the good *j*. Grouping the introduced quantities into vectors, the total endowment vector of agent *i* is $\mathbf{w}_i=(\mathbf{w}_{i1},\ldots,\mathbf{w}_{in})$, the total consumption vector of agent *i* is $\mathbf{x}_i=(\mathbf{x}_{i1},\ldots,\mathbf{x}_{in})$ and the price vector is $\mathbf{p}=(\mathbf{p}_1,\ldots,\mathbf{p}_n)$. Let $\mathbf{u}_i(\mathbf{x}_i)$: $\mathbf{R}_+^n \rightarrow \mathbf{R}_+$, describe the preference of agent *i* for different bundles of goods. At given prices, each agent sells their initial endowment, and buys a bundle of goods, which maximizes $\mathbf{u}_i(\mathbf{x}_i)$ subject to her budget constraints.

An equilibrium is a set of market clearing prices $\mathbf{p}=(\mathbf{p}_1,\ldots,\mathbf{p}_n)$ such that for agent *i*, there is a bundle of goods $\mathbf{x}_i=(\mathbf{x}_{i1},\ldots,\mathbf{x}_{in})$ such that the following two conditions hold:

i. For each agent *i*, the vector $\mathbf{x}_i$ maximizes $\mathbf{u}_i(\mathbf{x}_i)$ subject to the constraints

$$\sum_{k=1}^{n} \mathbf{p}_k * \mathbf{x}_{ik} \leq \sum_{k=1}^{n} \mathbf{p}_k * \mathbf{w}_{ik} \tag{1}$$

ii. For each good *j*,

$$\sum_{k=1}^{m} \mathbf{x}_{kj} \leq \sum_{k=1}^{m} \mathbf{w}_{kj} \tag{2}$$

According to Arrow and Debreu Theorem [5], such equilibrium exists under very mild conditions if the utility functions are concave. The First Fundamental Theorem asserts that such equilibrium is pareto efficient [5]. Fisher [4] in 1891 independently modeled a market, which consists of a set of **m** buyers and a set of **n** divisible goods. Each buyer *i* has an amount $\mathbf{e}_i$ of money and each good *j* has an amount $\mathbf{b}_j$ of this good. The preferences of agent *i* for different bundles of goods is denoted by $\mathbf{u}_i(\mathbf{x}_i)$: $\mathbf{R}_+^n \rightarrow \mathbf{R}_+$, where $\mathbf{x}_i=(\mathbf{x}_{i1},\ldots,\mathbf{x}_{in})$ is the consumption vector of agent *i*, and $\mathbf{x}_{ij}$ is the consumption of good *j* by agent *i*. Equilibrium prices is an assignment of prices of prices $\mathbf{P}=(\mathbf{p}_1,\ldots,\mathbf{p}_n)\in \mathbf{R}_+^n$ to the goods such that the market clears i.e., there is neither shortage nor surplus. In other words the following two conditions should hold:

i. For each buyer *i*, the vector $\mathbf{x}_i$ maximizes $\mathbf{u}_i(\mathbf{x}_i)$ subject to the constraints

$$\sum_{k=1}^{n} \mathbf{p}_k * \mathbf{x}_{ik} \leq \mathbf{e}_i \tag{3}$$

ii. For each good *j*,

$$\sum_{k=1}^{m} \mathbf{x}_{kj} \leq \mathbf{b}_j \tag{4}$$

In this study, load balancing problem is translated to Fisher's market model, where buyers are users and goods are computing resources. In the next section we discuss about the recursive competitive equilibrium.

3 Recursive Competitive Equilibrium Theory

Recursive Competitive Equilibrium Theory was first developed by Mehra and Prescott [11] and further refined in Prescott and Mehra [12] which establish the existence of, and asserts the pareto optimality of recursive competitive equilibrium. Recursive Competitive Equilibrium approach is one way of modeling uncertain dynamic phenomena to search for optimal actions.

Now, we define the dynamic case of Fisher's market model which consists of a set of **m** buyers and a set of **n** divisible goods. Let $\mathbf{e}_i(\mathbf{t})$ be the amount of money agent *i* has at time **t**, and $\mathbf{x}_i(\mathbf{t})=(\mathbf{x}_{i1}(\mathbf{t}),\ldots,\mathbf{x}_{in}(\mathbf{t}))$ be the consumption vector of agent *i* at time **t**, where $\mathbf{x}_{ij}(\mathbf{t})$ is consumption of good *j* by *i* at time **t**. Let $\mathbf{b}_j(\mathbf{t})$ be the amount of good *j* present at time **t**, and $\mathbf{p}_j(\mathbf{t})$ be the price of good *j* at time **t**. Let the preference of agent *i*, relative to the consumption $\mathbf{x}_i(\mathbf{t})$ at time **t** be denoted by the utility function $\mathbf{u}_i(\mathbf{t}, \mathbf{x}_i(\mathbf{t}))$.

At the start of each period **t**, the amount $\mathbf{e_i(t)}$ is determined from the previous period **t-1** as follows

$$\mathbf{e_i(t) = e_i(t-1) - \sum_{k=1}^{n} p_k(t-1) * x_{ik}(t-1)} \tag{5}$$

and the amount $\mathbf{b_j(t)}$ of good *j* at time **t** is determined as follows

$$\mathbf{b_j(t) = b_j(t-1) - \sum_{k=1}^{m} x_{kj}(t)} \tag{6}$$

Next, in each period, equilibrium prices $\mathbf{p=(p_1(t),\ldots,p_n(t))}$ are determined such that there is a bundle of goods $\mathbf{x_i(t)=(x_{i1}(t),\ldots,x_{in}(t))}$ and the following conditions hold:

i. For each buyer *i*, the vector $\mathbf{x_i(t)}$ maximizes $\mathbf{u_i(t,x_i(t))}$ subject to the constraints

$$\mathbf{\sum_{k=1}^{n} p_k * x_{ik}(t) \le e_i}(t) \tag{7}$$

ii. For each good *j*

$$\mathbf{\sum_{k=1}^{m} x_{kj}(t) \le b_j(t)} \tag{8}$$

Note that, the period prices depend only on the state variables in that period.

4 Proposed Distributed System Model

We consider a distributed system of **n** heterogeneous nodes (computing resources) connected by a communication network shared by **m** users. The terminology, notations, and assumptions used are similar to [14]. The job arrival rate of user *j* job at node *i* is $\mathbf{\Phi_i^j}$. Total arrival rate of user *j* jobs is $\mathbf{\Phi^j} = \mathbf{\sum_{k=1}^{n} \Phi_k^j}$. All the jobs in the system are assumed to be same. The service rate of node *i* is $\boldsymbol{\mu_i}$.

Out of user *k* jobs arriving at node *i*, the ratio $\mathbf{x_{ij}^k}$ of jobs is forwarded upon arrival through the communication means to another node (*j≠i*) to be processed there. The remaining ratio $\mathbf{x_{ii}^k = 1 - \sum_{j\neq i} x_{ij}^k}$ is processed at node *i*. That is, the rate $\mathbf{\Phi_i^k x_{ij}^k}$ of user *k* jobs that arrive at node *i* are forwarded through the communication means to node *j*, while the rate $\mathbf{\Phi_i^k x_{ii}^k}$ of user *k* jobs are processed at arrival node. Therefore, a set of values $\mathbf{x_{ij}^k}$ (for k=1,…,m; i=1,…,n; j=1,…,n) are to be chosen such that

$$\mathbf{\sum_{j=1}^{n} x_{ij}^k = 1} \qquad \mathbf{\forall\, i = 1,\ldots,n; k = 1,\ldots,m} \tag{9}$$

$$\mathbf{x_{ij}^k \ge 0} \qquad \mathbf{\forall\, i = 1,\ldots,n; j = 1,\ldots,n; k = 1,\ldots,m} \tag{10}$$

$$\sum_{k=1}^{m}\sum_{j=1}^{n}\Phi_j^k * x_{ji}^k < \mu_i \qquad \forall i = 1,\dots,n \tag{11}$$

Let us group these quantities into vectors as $x_i^k = (x_{i1}^k,\dots,x_{in}^k)$, $x^k = (x_1^k,\dots,x_n^k)$ and $x = (x^1,\dots,x^m)$.

Modeling each node as an M/M/1 queuing system [6], the expected node delay at node *i* is as follows

$$F_i(x) = \frac{1}{\mu_i - \beta_i} \tag{12}$$

where β_i is the load on node *i* and given as

$$\beta_i = \sum_{k=1}^{m}\sum_{j=1}^{n}\Phi_j^k * x_{ji}^k \tag{13}$$

It is to be noted that $F_i(x)$ is a strictly increasing, convex, and continuously differentiable function of X^j.

Let us assume that the expected communication delay of forwarding user *k* jobs at node *i* to node *j* is independent of two nodes but dependent on the total traffic through the network. Examples of such a case are local area networks and satellite communication systems, where the communication delay between any two nodes (or stations), depends on the total traffic generated by all nodes (or stations).

Let the traffic through the network be denoted by λ, where $\lambda = \sum_{j=1}^{m}\lambda^j$ and λ^j, is the traffic through the network due to user *j* jobs given as follows

$$\lambda^j = \frac{1}{2}\sum_{i=1}^{n}\left|\Phi_i^j - \beta_i^j\right| \tag{14}$$

where $\beta_i^j = \sum_{k=1}^{n}\Phi_k^j x_{ki}^j$ is the contribution on the load on node *i* by user *j* jobs.

Modeling the communication network as an M/M/1 queuing system [6], the expected communication delay of any job is given as

$$G(\lambda) = \frac{t}{1 - t\sum_{k=1}^{m}\lambda^k} \tag{15}$$

where **t** is the mean communication time for sending and receiving a job from one node to the other for any user. Clearly, $G(\lambda)$ is a positive, non-decreasing, convex, and continuously differentiable function of λ.

Therefore, over all response time of user *j* job is the sum of expected node delay at each node *i* and expected communication delay given as follows

$$T^j(x) = \frac{1}{\Phi^j}\sum_{i=1}^{n}\beta_i^j F_i(x) + \frac{\lambda^j}{\Phi^j}G(\lambda) \tag{16}$$

The mean response time of all jobs is given by

$$\mathbf{T(x)} = \frac{\mathbf{1}}{\mathbf{\Phi}} \sum_{\mathbf{j=1}}^{\mathbf{m}} \mathbf{\Phi^j T^j(x)} \tag{17}$$

The best response time for user *j* job is a solution to the following optimization problem

$$\min_{\mathbf{x^j}} \mathbf{T^j(x)} \tag{18}$$

subject to the constraints (9) to (11).

5 Competitive Equilibrium Approach for Static Load Balancing

Let us review the scheme described in [8]. At first the distributed system model described in the previous section is translated to Fisher's market model, where buyers are users and goods are computing resources. Each user *j* is endowed a monetary budget $\mathbf{w_i} \geq 0$ to purchase computing power, and has utility function $\mathbf{U^j(x) = -T^j(x)}$ to denote her preferences for various computing resources. The price for executing unit job at node *i* is $\mathbf{p_i}$.

The competitive equilibrium solution to load balancing is to find a set of prices and allocation of jobs to computing resources such that each user maximizes her utility subject to her budget constraints, and the market clears i.e.,

$$\max_{\mathbf{x^j}} \mathbf{u^j(x)} \qquad \forall\, j = 1,\ldots,m \tag{19}$$

subject to the constraints (9) to (11) and market clearing condition given by

$$\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{p_i * \beta_i^j \leq w_j} \qquad \forall\, j = 1,\ldots,m \tag{20}$$

where $\mathbf{u^j(x)}$ is strictly continous, concave, and continuosly differentiable function of $\mathbf{x^j}$. Also $\mathbf{x^j \subseteq R_+^n}$ is a closed convex set bounded from below. According to Arrow and Debreu [4] the necessary and sufficient conditions for the existence of competitive equilibrium are satisfied. Hence there exists a competitive equilibrium for the given load balancing problem.

We present below the algorithm CES (Competitive Equilibrium Solution) for computing equilibrium prices and load allocation of jobs at various resources.

5.1 Algorithm CES

```
Input
   Node Processing Rates: $\mu_1,\ldots,\mu_n$
       Job Arrival Rates: $\Phi_i^j \forall j=1,...,m; i=1,...,n$
Output
    Load Fractions $x^1,\ldots,x^m$
```

```
1. Intialization
      1.1.  $w_j \rightarrow 1$        $\forall j=1,\dots,m$
      1.2.  $p_i \rightarrow 1/n$      $\forall i=1,\dots,n$
2. Loop
        2.1. At prices $p_1,\dots,p_n$ compute $x^1,\dots,x^m$ such that
             each user maximizes her utility function
             (19) subject to the constraints (9) to (11)
        2.2. Obtain market clearing error, $\alpha$ given as
             follows
```

$$\alpha = \sqrt{\sum_{j=1}^{m} \xi_j^2} \tag{21}$$

```
             where $\xi_j$ is given by
```

$$\xi_j = w_j - \sum_{i=1}^{n} p_i * \beta_i^j \tag{22}$$

```
        2.3. Adjust the prices $p_1,\dots,p_n$ in proportion to
             aggregate demands
   Until $\alpha \le$ error tolerance
```

This is an artificial trade, where price **p** and budget **w** do not have any physical interpretations and have no outside use. They are only an economic means for achieving individual and system optimality. The meaningful output of our problem is only the load distribution.

5.2 Complexity of Algorithm CES

According to Arrow and Debreu [4] the load balancing problem is guaranteed to have competitive equilibrium. Hence the algorithm CES is PPAD-Complete whose convergence is unpredictable but there surely exists a solution.

6 Dynamic Load Balancing

In this section, we extend the static load balancing scheme described in the previous section.

In general, a decentralized load balancing scheme has three components.

i. **an information policy** used to exchange state information (number of jobs waiting in the queue to be processed) between the nodes every P time units.
ii. **a transfer policy** that determines whether the job should be processed locally or transferred to another node for processing.
iii. **a location policy** that determines the destination node for remote processing.

6.1 Recursive Competitive Equilibrium Approach for Dynamic Load Balancing

Let $\mathbf{N}_i^j$ denote the mean number of user *j* jobs at node *i* and $\mathbf{r}_i$ denote the mean service time of a job at node *i*. The user *j* marginal node delay is defined as

$$\mathbf{f_i^j(\beta_i) = \frac{\partial}{\partial \beta_i^j}(\beta_i^j F_i^j(\beta_i)) = \frac{(\mu_i - \sum_{k=1,k\neq j}^{m} \beta_i^k)}{(\mu_i - \sum_{k=1}^{m} \beta_i^k)^2}} \tag{23}$$

where $\mathbf{\beta_i^k F_i^k(\beta_i)}$ denotes the mean number of user *k* jobs at node *i*.

Rewriting the above equation, in terms of $\mathbf{r_i}$ and $\mathbf{N_i^j}$

$$\mathbf{f_i^j(\beta_i) = r_i(1 + \sum_{k=1}^{m} N_i^k)(1 + N_i^j)} \tag{24}$$

Let $\mathbf{\rho = t \sum_{k=1}^{m} \lambda^k}$, denote mean utilization of the communication network, and $\mathbf{\rho^{-j} = t \sum_{k=1,k\neq j}^{m} \lambda^k}$ denote the utilization of communication network excluding user *j* traffic. The marginal communication delay of user *j* job is defined as

$$\mathbf{g^j(\lambda) = \frac{\partial}{\partial \lambda^j}(\lambda^j G^j(\lambda)) = \frac{t(1 - t \sum_{k=1,k\neq j}^{m} \lambda^k)}{(1 - t \sum_{k=1}^{m} \lambda^k)^2}} \tag{25}$$

where $\mathbf{\lambda^k G^k(\lambda)}$ denotes the mean number of user jobs in the communication network. Denoting the above equation in terms of ρ and ρ^{-j} we have

$$\mathbf{g^j(\lambda) = \frac{t(1-\rho^{-j})}{(1-\rho)^2} \qquad \rho < 1} \tag{26}$$

Let $\mathbf{\rho'}$ denote the utilization of the network at a given instant, and $\mathbf{\rho^{-j\prime}}$ denote the utilization of the network at a given instant excluding user *j* traffic. Similarly, let $\mathbf{n_i^j}$ denote the number of user *j* jobs at *i*, at a given instant. Expressing $\mathbf{f_i^j(\beta_i)}$ and $\mathbf{g^j(\lambda)}$ in terms of instant variables, we have

$$\begin{aligned} &\mathbf{f_i^j = r_i(1 + \sum_{k=1}^{m} n_i^k)(1 + n_i^j)} \\ &\mathbf{g^j = \frac{t(1-\rho^{-j\prime})}{(1-\rho')^2} \qquad \rho' < 1} \end{aligned} \tag{27}$$

We now describe the three components of dynamic load balancing in detail

i. **Information Policy**-Every node broadcasts its state (i.e., queue length $\mathbf{n_i^j}$) to all other nodes every **P** time units.
ii. **Transfer Policy**-A threshold is used to determine whether user *j* job arriving at node *i* should be processed locally or transferred to another node.

Let $\mathbf{T}_i^j$ be the threshold at node *i* for user *j* job. Each node *i* broadcasts its arrival rate $\boldsymbol{\Phi}_i^j$ to all other nodes every **P1**time units, where **P1>>P**. From this information all the nodes determine the optimal loads $\boldsymbol{\beta}_i^j$ using the algorithm **CES**. Threshold $\mathbf{T}_i^j$ is the optimal number of user *j* jobs that can be present at node *i*. ($\mathbf{T}_i^j = \boldsymbol{\beta}_i^j \mathbf{F}_i^j(\boldsymbol{\beta}_i)$)

When the number of jobs of user *j* at node *i* is greater than $\mathbf{T}_i^j$, then it is eligible for transfer.

iii. **Location Policy**-The destination node for user *u* job that is eligible for transfer is determined as follows.

   a. First, a node with the lightest load for user *u* job is determined. If $\mathbf{f}_i^u > \mathbf{f}_j^u + \mathbf{g}^u$, then node *j* is lightly loaded than node *i* for user *u* job. Let $\boldsymbol{\delta}_i^u = \max\limits_{\mathbf{j}} (\mathbf{f}_i^u - (\mathbf{f}_j^u + \mathbf{g}^u))$. If $\boldsymbol{\delta}_i^u$ >0, then node *j* is the lightest loaded node for user *u* job, otherwise there is no light node and user *u* job should be processed locally.
   b. Let **c** denote the number of times that a user *u* job has been transferred. Let $\boldsymbol{\omega}$ ($0<\omega\leq 1$) be a weighting factor used to prevent a job from being transferred many times. And let Δ (Δ>0) be a bias used to protect the system from instability by not allowing the load balancing policy to react to small changes. If $\boldsymbol{\omega}^c\ \boldsymbol{\delta}_i^u > \Delta$, then the job of user *u* will be transferred to node *j*. Otherwise it will be processed locally.

7 Experimental Results

A computer model is run to evaluate the effects of different schemes on mean response time of all jobs and individual response time of each job. The proposed dynamic scheme (RCES), the static load balancing scheme using competitive Equilibrium (CES), and Nash equilibrium scheme (NES) are implemented for comparison purposes.

The parameters used for the experiments are given below:

i. The distributed system consists of 16 computers with service rates as shown in Table 1

Table 1. Service Rates of the Computers

Computer	1-6	7-11	12-14	15-16
Service Rate (jobs/sec)	10	20	50	100

ii. The system has 10 users with job arrival fractions $\mathbf{q^j}$ as given in Table 2. The actual arrival rate $\mathbf{\Phi^j}$ of user *j* is calculated to give the required overall system load $\boldsymbol{\rho}$ and is given by

$$\Phi^j = q^j * \rho * \sum_{j=1}^{n} \mu_j \tag{28}$$

The job arrival rates of each user *j*, j=1,...,m to each computer *i*, i=1,…,n i.e., Φ_i^j are obtained as

$$\Phi_i^j = q^j * \mu_i \tag{29}$$

Table 2. Job Arrival Fractions of Each User

User	1	2	3-6	7-9	10
Job Arrival Fractions q^j	0.3	0.2	0.1	0.001	0.07

iii. The mean communication time for a job **t**, is set to 0.01 sec. The communication overhead **OV** is the percentage of service time that a computer has to spend to send or receive a job. The over head value is set to 0%, 5% and 10% to show the effect of communication overhead on the average mean response time and individual response time.

7.1 Nash Equilibrium Scheme (NES)

In this scheme, each user *j* (j=1,…,m) must find the load assigned to each computing resource *i* (i=1,…,n) such that the response time of his own jobs is minimized. The best response of user *j* job is a solution to the optimization problem given by (18).

The algorithm for NES is run by initializing strategy $\mathbf{x^i}$ of each player *i* to zero vector. Each player then updates its strategy $\mathbf{x^i}$ in a sequential manner by solving the optimization problem (18). An interesting case occurs when no player can change its strategy $\mathbf{x^{i*}}$, and decrease its response time by choosing a different strategy $\mathbf{x^{i*}}$ when the other users' strategies are fixed. In this case, the system is said to reach Nash equilibrium.

7.2 Effect of System Utilization

Figures 1, 2, and 3 present the effect of system utilization (ranging from 10% to 90%) on the mean response time of all users when communication overhead OV is 0%, 5% and 10% respectively. The bias for job transfer ($\boldsymbol{\Delta}$) is set to 0.4, the exchange period of state information (**P**) is set to 0.1 sec., and the weighting factor for job transfer ($\boldsymbol{\omega}$) is set to 0.9 in all the cases.

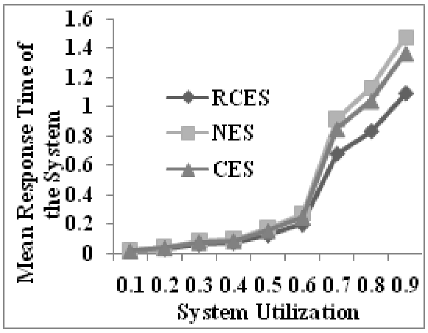


Fig. 1. System Utilization Vs Mean Response Time of the System when OV=0%

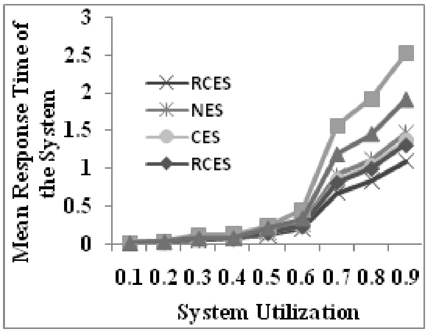


Fig. 2. System Utilization Vs Mean Response Time of the System when OV=5%

In figure 1, and 2 when OV is 0%, and 5%, it can be observed that at low system utilization (load), all the schemes show similar performance. But with the increase in system utilization, RCES yields higher performance than both CES and NES. However, in figure 3, when OV is 10%, the performance of RCES is insensitive to overheads at low to medium system loads. But at high system loads, the performance of RCES degrades and the static schemes are more efficient because of their less complexity.

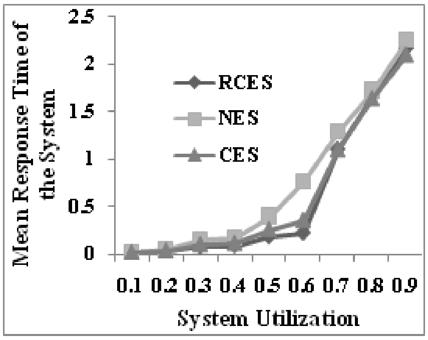


Fig. 3. System Utilization Vs Mean Response Time of the System when OV=10%

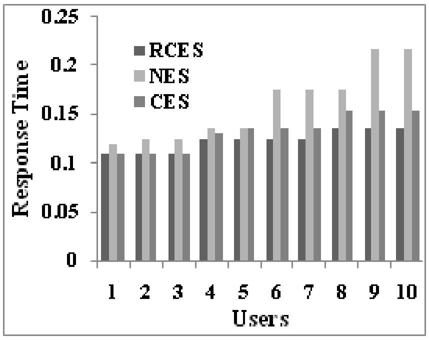


Fig. 4. Response Time of Each User when OV=5% and System Utilization=50%

Figure 4, presents the response time of each user when system utilization is 50%, and communication overhead is 5%. The other parameters, **Δ**, **ω**, and **P** are all fixed as before. It can be observed that the differences in response times for RCES is very small compared to CES and NES.

7.3 Effect of Bias

In figure 5, we present the variation of mean response time with system utilization for various biases. The overhead is assumed to be 5%. The other parameters are fixed

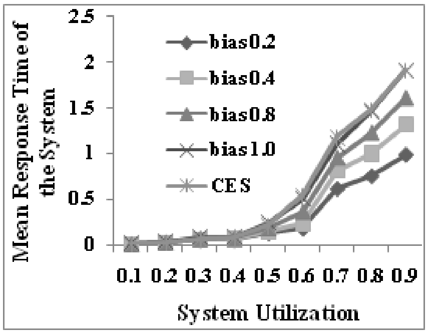


Fig. 5. Effect of Bias on Mean Response Time (OV=5%)

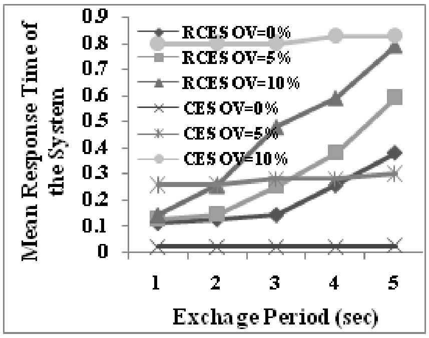


Fig. 6. Effect of Exchange Period on Mean Response Time (System Utilization=50%)

(**P**=0.1sec, **ω**=0.9). It can be observed that as the bias increases, the expected response time of RCES increases, and for a high bias (**Δ**=1), the performance of RCES is similar to CES.

7.4 Effect of Exchange Period

In figure 6, we present the variation of mean response time with exchange period of system state information. The system utilization is assumed to be 50%. The other parameters are fixed (**Δ**=0.4, **ω**=0.9). It can be observed that the mean response time of RCES increases with increase in exchange period, because for high values of P, outdated state information is exchanged between the nodes and optimal load balancing will not be done.

8 Conclusions

Our study proposes recursive competitive equilibrium approach for dynamic load balancing a computational grid. A computer model of a grid is run with various system loads and communication overheads. Two other schemes- competitive equilibrium scheme and Nash equilibrium scheme are implemented. It was observed that, at low communication overheads, RCES yields superior performance over CES and NES. Furthermore, RCES is fairer than both CES and NES as it provides almost equal response times for all the users. Moreover, as the bias, exchange period and overheads for communication are increased, CES and RCES yield similar performance.

References

1. Grosu, D., Chronopolous, A.T.: Non Cooperative Load Balancing in Distributed Systems. Journal of Parallel and Distributed Computing 65(9), 1022–1034 (2005)
2. Grosu, D., Chronopoulous, A.T., Leung, M.Y.: Cooperative Load Balancing in Distributed systems. Concurrency and Computation: Practices and Experience 20(16), 1953–1976 (2008)

3. El-Zoghdy, S.F., Kameda, H., Li, J.: A comparative Study of Static and Dynamic Individually Optimal Load Balancing Policies. In: The Proceedings of IASTED International Conference on Networks, Parallel, and Distributed Processing and Applications (2002)
4. Scarf, H.: The Computation of Economic Equilibria. In: Cowles Foundation Monograph, vol. 24, Yale University Press, New Haven (1973)
5. Arrow, K.J., Debreu, G.: Existence of an Equilibrium for a Competitive Economy. Econometrica 22(3), 265–290 (1954)
6. Kleinrock, L.: Queuing Systems-Theory, vol. 1. John Wiley and Sons, Chichester
7. Shahu Chatrapati, K., Ujwala Rekha, J., Vinaya Babu, A.: Competitive Equilibrium Approach for Load Balancing in Computational Grids. In: The Proceedings of the International Conference on Advances and Emerging Trends in Computing Technologies (2010)
8. Shahu Chatrapati, K., Ujwala Rekha, J., Vinaya Babu, A.: Competitive Equilibrium Approach for Load Balancing a Computational Grid with Communication Delays. Journal of Theoretical and Applied Information Technology 19(2), 126–133 (2010)
9. Youran, L.: A Dynamic Load Balancing Mechanism for Distributed Systems. Journal of Computer Science and Technology 11(3), 195–207 (1996)
10. Walras, L.: Elements of Pure Economics; or the Theory of Social Wealth, Lausanna, Paris (1874)
11. Mehra, R., Prescott, E.C.: Recursive Competitive Equilibria and Capital Asset Pricing. In: Mehra, R. (ed.) Essays in Financial Economics, Doctoral Dissertation, Carnegie Mellon University, UMI, Ann Arbor, Michigan (1977)
12. Prescott, E.C., Mehra, R.: Recursive Competitive Equilibria: The Case of Homogeneous Households. Econometrica 48, 1365–1379 (1980)
13. Subrata, R., Zomaya, A.Y.: Game Theoretic Approach for Load Balancing in Computational Grids. IEEE Transactions on Parallel and Distributed Systems 19(1), 66–76 (2008)
14. Penmatsa, S., Chronopolous, A.T.: Dynamic Multi-User Load Balancing in Distributed Systems. In: The Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium (2006)
15. Spata, M.O.: A Nash Equilibrium Based Algorithm for Scheduling Jobs in a Grid Cluster. In: The Proceedings of 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, vol. 18(20), pp. 251–252 (2007)
16. Tang, X., Chason, S.T.: Optimizing Static Job Scheduling in a Network of Heterogeneous Computers. In: The Proceedings of International Conference on Parallel Processing (2000)
17. Zeng, Z., Veeravalli, B.: Design and Performance Evaluation of Queue-and-Rate-Adjustment Dynamic Load Balancing Policies for Distributed Networks. IEEE Transactions on Computers 55(11), 1410–1422 (2006)

Smoothed Functional and Quasi-Newton Algorithms for Routing in Multi-stage Queueing Network with Constraints

K. Lakshmanan and Shalabh Bhatnagar

Department of Computer Science and Automation,
Indian Institute of Science, Bangalore 560012, India
{lakshmanank,shalabh}@csa.iisc.ernet.in

Abstract. We consider the problem of optimal routing in a multi-stage network of queues with constraints on queue lengths. We develop three algorithms for probabilistic routing for this problem using only the total end-to-end delays. These algorithms use the smoothed functional (SF) approach to optimize the routing probabilities. In our model all the queues are assumed to have constraints on the average queue length. We also propose a novel quasi-Newton based SF algorithm. Policies like Join Shortest Queue or Least Work Left work only for unconstrained routing. Besides assuming knowledge of the queue length at all the queues. If the only information available is the expected end-to-end delay as with our case such policies cannot be used. We also give simulation results showing the performance of the SF algorithms for this problem.

1 Introduction

Multi-stage queueing networks are used for modelling manufacturing systems. They also find applications in telecommunication networks, multi-stage interconnection networks [11] and server farms [9]. Constraints are ubiquitous in real world systems, like QoS routing in the Internet [15] or energy constraints in sensor networks [1]. Queueing network models have been used to analyze these systems. Analyzing a general queueing network is hard and can involve complications like Braess paradox [6]. Policies like Join the Shortest Queue(JSQ), Least Work Left(LWL), Central-Queue-Shortest-Job (CQSJ) or Size-Interval Splitting are used for task assignment (routing) when there are multiple parallel queues [10]. Theoretical analysis of these policies is hard. Another drawback is that these policies are known to work only for unconstrained routing.

In many optimization problems the relationship between the parameters and the objective function is not explicitly known. The objective itself may be a parameterized long-run average of a certain sample cost function which has to be estimated by simulation. In the case of gradient search algorithms one is often interested in finding the zeros of an objective function that corresponds to the gradient of average cost. It is not possible in most cases to calculate the gradient of such an objective analytically. Simulation has been widely used in such scenarios.

Amongst the algorithms that estimate the gradient,the Simultaneous Perturbation Stochastic Approximation (SPSA) [14] is popular. This is because it uses only two simulations at any instant regardless of the parameter dimension. Another gradient estimation technique with good performance is based on the idea of randomly perturbing parameters and goes by the name smoothed functional (SF) scheme. It is originally due to Katkovnik and Kulchitsky [12]. Here the idea is to approximate the gradient by a convolution with a multi-normal distribution. In Bhatnagar [2], a similar technique was used to estimate the second derivative as well. A problem of constrained optimization with long-run average cost objective and constraint functions is considered in Bhatnagar et al. [3]. Convergence of these SF algorithms has also been proved in [2] and [3].

A multi-stage shortest path problem is considered in Kolavali and Bhatnagar [13]. Here the links between nodes are assumed to have different weights. However there are no constraints considered. As in our model the link weights are not known to the algorithm, and the only information available is the total path length for any particular path chosen. The objective is to find the shortest path. Algorithms based on Q-Learning and Multi-Agent Foraging Ant Colony Optimization (MAF-ACO) [5] are used. The MAF-ACO algorithm in [5] is similar to Ant Colony Optimization (ACO) algorithms [8] and relies on the differential path length effect. But unlike ACO algorithms it does not use any complex heuristics or elitist policies. This makes the algorithm suitable for network routing. Convergence of these algorithms is studied in [5] using stochastic approximation theory.

In this paper we consider the problem of routing in multi-stage network of queues with constraints on queue lengths. Instead of link weights, the queueing delays are considered here as they capture the real networks better. To the best of our knowledge this is the first time that this problem is considered. We propose three algorithms which route traffic probabilistically based on the total end-to-end delay. Unlike in policies used for task assignment like JSQ,CQSF,LWL we do not use information on individual queue lengths that vary rapidly. Further these policies determine the next queue in the path for individual packets, while our algorithms tune only the routing probabilities.

In dynamic problems like routing it is not possible to directly estimate the gradient or Hessian of the cost analytically. Hence the smoothed functional method is used. In a Newton based algorithm we need to evaluate the inverse of the Hessian matrix which can be expensive. We therefore develop for the first time the quasi-Newton smoothed functional (QN-SF) algorithm that does not require the computation of the Hessian inverse. Our algorithm is based on the popular quasi-Newton algorithm BFGS [7] for approximating the inverse of the Hessian.

The rest of the paper is organized as follows. Section 2 describes the multi-stage constrained routing problem and the framework of constrained optimization. The next section describes the G-SF and N-SF algorithms for the routing problem. In section 4 quasi-Newton methods are briefly explained and the QN-SF algorithm is presented. Section 5 discusses the simulation results. Finally concluding remarks are given in section 6.

2 Routing in Multi-stage Queueing Network - Problem Formulation

Routing in communication networks is a dynamic problem where queue lengths and delays change with the traffic. Simulation is often used to study these systems as analyzing such networks theoretically is hard in general. Besides it also often involves constraints involving delays or bandwidth.

We model our system as a multi-stage network of queues as in Fig. 1. There is no assumption on service time or arrival distribution. There is a queueing delay at each node. The packets have to be routed from the source node s to destination d. Here the objective is to find a path that minimizes the total delay from source to destination. Further this path should be such that it satisfies certain constraints on the average queue length at each node. Thus the problem we are interested in can be posed as a constrained optimization problem.

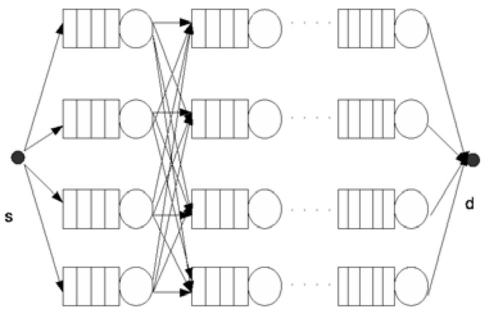


Fig. 1. Network of queues

Let the total number of queues be N. Each stage has a fixed number M of queues and it is assumed that all queues in one stage are connected to the next stage. Hence each queue at every stage except the last one is connected to M other queues and similarly except the first stage, each queue is fed by M other queues. The total dimension of routing probability is thus $K = (N-M+1)*M$.

The parameter $\theta(n)$ to be tuned is the routing probability vector $\boldsymbol{p}(n) = [p_l(n),\, l = 1,\ldots,K]^T$ at the queues. Let $p_{(i*j)}(n)$, $i = 1,\ldots,(N-M+1)$ and $j = 1,\ldots,M$, $i*j = l$, be the probability of selecting choice j, $0 < j \leq M$ after service of a packet at the node i. Given a routing probability vector $\theta(n) = \boldsymbol{p}(n)$, the vector of queue lengths at the nodes $q(n) = [q_i(n),\ldots,q_N(n)]^T$ is Markov. Hence $\{q(n)\}$ is a parameterized Markov process.

The queueing delay incurred at node i is $h_i(q(n)) = E\big[\text{delay at } i \big| q(n), \boldsymbol{p}(n)\big]$. The total delay $h\big(q(n)\big)$ is the expected end-to-end delay for a packet, given the vector of queue lengths $q(n)$ and vector routing probabilities $\boldsymbol{p}(n)$, and is given by

$$h\big(q(n)\big) = \sum_{i=1}^{N-M+1} \sum_{j=1}^{M} \Big(p_{(i*j)}(n) * h_j\big(q(n)\big)\Big)$$

The long-run average delay $J(\theta)$ from the source to the destination node is

$$J(\theta) = \lim_{n\to\infty} \frac{1}{n} \sum_{j=0}^{n-1} h\big(q(j)\big) \tag{2.1}$$

The constraint function $g_i\big(q(n)\big) = E\big[q_i(n)\big|\boldsymbol{p}(n)\big]$, $\forall i = 1,\ldots,N$ at node i is the expected queue length at that node. There are constraints on the long-run average queue length at each node

$$G_i(\theta) = \lim_{l\to\infty} \frac{1}{l} \sum_{j=0}^{l-1} g_i\big(q(j)\big) \leq \alpha_i,\ \forall i = 1,\ldots,N$$

The objective thus is to minimize the long-run expected queue length $J(\theta)$ such that the constraints $G_i(\theta) \leq \alpha_i$ are satisfied for certain prescribed thresholds α_i, $i = 1,\ldots,N$. The SF algorithms are used to update probability vector $p_{(i*j)}$ at a node i. To ensure that the probabilities at each queue add up to 1 after each update, the normalized probability $\hat{p}_l = \frac{p_l}{\sum_l p_l}$ is used.

Note that the tunable parameter θ takes values in the constraint set $C = [0,1]^N$ which is closed and bounded (hence compact) set. Our aim is to find the optimum $\theta^* \in C$ s.t.

$$\theta^* = \arg\min\{J(\theta) \mid \theta \in C,\ G_i(\theta) \leq \alpha_i,\ i = 1,2,\ldots,p\}$$

We assume that there exists at least one $\theta \in C$ for which all the above constraints are satisfied. For given $\lambda_1, \lambda_2, \ldots, \lambda_N \in \mathbb{R}^+ \cup \{0\}$ let

$$L(\theta, \lambda_1, \lambda_2, \ldots, \lambda_N) = J(\theta) + \sum_{i=1}^{N} \lambda_i (G_i(\theta) - \alpha_i)$$

denote the Lagrangian. Let $\boldsymbol{\lambda}$ denote the vector $(\lambda_1, \lambda_2, \ldots, \lambda_N)^T$.

The following assumptions are required to prove convergence of our algorithms, see [2] for details of convergence of smoothed functional algorithms in a general setting.

Assumption A1. The functions $J(.)$ and $G_i(.), i = 1,2,\ldots,N$ are twice continuously differentiable.

Assumption A2. The functions $h(.)$ and $g_i(.)$, $i = 1,\ldots,N$ are Liptschitz continuous.

In the case of Newton based algorithms, the Hessian estimate is projected onto the space of positive definite and symmetric matrices after each iteration. This is done to ensure that the algorithm proceeds along the negative gradient direction.

Let $P : \mathbb{R}^N \to \{\text{positive definite and symmetric matrices}\}$ denote the projection operator. If A is positive definite and symmetric then $P(A) = A$. We assume the operator P satisfies the following condition.

Assumption A3. If $\{A_n\}$ and $\{B_n\}$ are sequences of matrices in $R^{N\times N}$ such that $\lim\limits_{n\to\infty} \|A_n - B_n\| = 0$ then $\lim\limits_{n\to\infty} \|P(A_n) - P(B_n)\| = 0$. Further for any sequence $\{C_n\}$ of matrices if $\sup\limits_n \|C_n\| < \infty$, then $\sup\limits_n \|P(C_n)\|, \sup\limits_n \|P(C_n)^{-1}\| < \infty$.

3 Smoothed Functional Algorithms for Constrained Routing

Gradient and Newton based SF algorithms for constrained optimization have been studied in [3]. For details and derivation of the SF scheme, see [2]. Let $\Gamma(x) = \big(\Gamma_1(x), \ldots, \Gamma_K(x)\big)^T$ represent the projection of $x \in \mathbb{R}^K$ on to the set $C = \Big\{x \in [0,1]^K \Big| \sum_{i=jm+1}^{(j+1)m} x_i = 1, \forall\, j = 0, 1, \ldots, N-M+1\Big\}$ which is a compact set. Let $\hat{\Gamma} : \mathbb{R} \to [0,\infty)^K$ denote the projection $\hat{\Gamma}(y) = \max(y, 0), y \in \mathbb{R}$.

3.1 Gradient SF Algorithm

This algorithm aims to update the parameter along the steepest descent direction. Let $\eta = \big(\eta_1, \eta_2, \ldots, \eta_K\big)^T$ be a vector of independent $\mathcal{N}(0,1)$-distributed random variables $\eta_1, \eta_2, \ldots, \eta_K$. The following gradient estimate is used here, see [2] for details.

$$\nabla_\theta L(\theta, \boldsymbol{\lambda}) = \lim_{\beta \to 0} E\Big[\frac{\eta}{\beta}(L(\theta + \beta\eta, \boldsymbol{\lambda}) - L(\theta, \boldsymbol{\lambda})\Big] \tag{3.1}$$

We use the G-SF algorithm to update the routing probability vector $\boldsymbol{p}(n) = [p_l(n), \forall l = 1, \ldots, K]^T$ and Lagrange parameter $\boldsymbol{\lambda}(n) = [\lambda_i(n), \forall i = 1, \ldots, N]^T$.

Define step-sizes sequences $\{a(n)\}$, $\{b(n)\}$ and $\{c(n)\}$ that satisfy the requirements

$$\sum_n a(n) = \sum_n b(n) = \sum_n c(n) = \infty; \sum_n a(n)^2, \sum_n b(n)^2, \sum_n c(n)^2 < \infty; \tag{3.2}$$

$$a(n) = o(b(n));\ b(n) = o(c(n)). \tag{3.3}$$

3.2 Newton-SF Algorithm

The Newton version of SF algorithm has been first studied in Bhatnagar [2]. Let $Z_{ij}(n)$, $i, j = 1, \ldots, N$, denote an estimate of the Hessian and let $H(n) = P\big([[Z_{ij}(n)]]_{i,j=1}^N\big)$ denote its projection to the space of symmetric and positive definite matrices. Let $M(n) = [[M_{i,j}(n)]]_{i,j=1}^N \triangleq H(n)^{-1}$ be the inverse of $H(n)$. In the Jacobi version that we implemented only the diagonal entries $H_{i,i}$ are considered so,

$$M_{i,j}(n) = \begin{cases} \frac{1}{H_{i,i}(n)} & i = j \text{ if } H_{i,i}(n) > 0, \\ 0 & i \neq j \end{cases}$$

For the case when $H_{i,i}(n) \leq 0$, we first set $H_{i,i}(n) = \epsilon$ for some $\epsilon > 0$ and then set $M_{i,i}(n) = 1/\epsilon$. In this algorithm we have an additional step-size $d(n)$ that satisfies $\sum_n d(n) = \infty, \sum_n d(n)^2 < \infty$ and $c(n) = o(d(n))$. Further step-sizes $a(n), b(n)$ and $c(n)$ satisfy (3.2) and (3.3). Convergence of both G-SF and N-SF algorithms for constrained optimization has been proved in [3]. It is also shown in the paper that the optimal point to which the algorithms converge is feasible i.e; the constraints are satisfied at the point of convergence.

Algorithm 1. G-SF algorithm for constrained routing

```
1: Initialize $Z_l = 0$, $p_l = 0.1$, $\forall l = 1, \ldots, K$. Fix $\mathcal{M}$ and small $\beta > 0$ and set $n = 0$.
2: while $n < \mathcal{M}$ do
3:     Generate i.i.d $\mathcal{N}(0,1)$ random variables $\boldsymbol{\eta}(n) = \big(\eta_1(n), \eta_2(n), \ldots, \eta_K(n)\big)^T$. Generate two parallel simulations of the queue $\{q(n)\}$ and $\{q'(n)\}$ governed by parameters $\boldsymbol{p}(n)$ and $\boldsymbol{p}(n) + \beta\boldsymbol{\eta(n)}$. For $l = 1, \ldots, K$ and $i = 1, \ldots, N$,
       $$Z_l(n+1) = Z_l(n) + c(n)\bigg(\frac{\eta_l(n)}{\beta}\Big(h\big(q'(n)\big) + \sum_{i=1}^{N} \lambda_i(n) Y_i'(n) - h\big(q(n)\big) - \sum_{i=1}^{N} \lambda_i(n) Y_i(n)\Big) - Z_l(n)\bigg) \quad (3.4)$$
       $$p_l(n+1) = \Gamma_l\big(p_l(n) - b(n) Z_l(n)\big) \quad (3.5)$$
       $$Y_i(n+1) = Y_i(n) + c(n)\big(q_i(n) - Y_i(n)\big) \quad (3.6)$$
       $$Y_i'(n+1) = Y_i'(n) + c(n)\big(q_i'(n) - Y_i'(n)\big) \quad (3.7)$$
       $$\lambda_i(n+1) = \hat{\Gamma}\big(\lambda_i(n) + a(n)(Y_i(n) - \alpha_i)\big) \quad (3.8)$$
4:     Set $n := n + 1$
5: end while
6: Output the routing probability $\boldsymbol{p}(n)$ and terminate.
```

4 Quasi-Newton Algorithms

The Newton's method for solving unconstrained minimization is in general cumbersome because in addition to computing the Hessian it is also required to solve a linear system of equations to find its inverse, see chapter 4 of [4]. This has led to a new class of algorithms known as quasi-Newton methods, variable metric or secant methods. The paper by Dennis and More [7] gives a detailed survey on these algorithms. These algorithms do not find the exact Hessian inverse but approximations of it at each iteration are used.

4.1 Quasi-Newton SF Algorithm

We briefly describe the popular **BFGS** (Broyden-Fletcher-Goldfarb-Shanno) update for optimization problems. For simplicity consider the following unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

The following is the general algorithm used for finding zeros of the gradient $\nabla f(x)$

$$x(n+1) = x(n) - \gamma(n) M(n) \nabla f\big(x(n)\big)$$

where $\gamma(n)$ is the step size and $M(n)$ is an approximation to the inverse of the Hessian. Let $s = s(n) = x(n) - x(n-1)$ and $u = u(n) = f'\big(x(n)\big) - f'\big(x(n-1)\big)$.

Algorithm 2. N-SF algorithm for constrained routing

```
1: Initialize $Z_l = 0$, $p_l = 0.1$, $\forall l = 1, \ldots, K$. Fix $\mathcal{M}$ and $\beta > 0$ and set $n = 0$.
2: while $n < \mathcal{M}$ do
3:     Generate i.i.d $\mathcal{N}(0,1)$ random variables $\boldsymbol{\eta}(n) = \big(\eta_1(n), \eta_2(n), \ldots, \eta_K(n)\big)^T$. Generate two parallel simulations $\{q(n)\}$ and $\{q'(n)\}$ of the queue governed by routing probability parameters $\boldsymbol{p}(n)$ and $\boldsymbol{p}(n) + \beta\boldsymbol{\eta}(n)$. For $j, k = 1, \ldots, K, j < k$,
       $$Z_{i,i}(n+1) = \big(1 - d(n)\big)Z_{i,i}(n) + d(n)\left(\frac{\eta_i^2(n) - 1}{\beta^2}\Big(h\big(q'(n)\big) + \sum_{i=1}^{N} \lambda_i(n)Y_i'(n) - h\big(q(n)\big) - \sum_{i=1}^{N} \lambda_i(n)Y_i(n)\Big)\right) \quad (3.9)$$
       $$Z_{j,k}(n+1) = \big(1 - d(n)\big)Z_{j,k}(n) + d(n)\left(\frac{\eta_j(n)\eta_k(n)}{\beta^2}\Big(h\big(q'(n)\big) + \sum_{i=1}^{N} \lambda_i(n)Y_i'(n) - h\big(q(n)\big) - \sum_{i=1}^{N} \lambda_i(n)Y_i(n)\Big)\right) \quad (3.10)$$
       And for $j > k$, set $Z_{j,k}(n+1) = Z_{k,j}(n+1)$.
4:     Use equations (3.4), (3.6), (3.7) and (3.8) for updating $Z_l, Y_i, Y_i'$ and $\lambda_i$ respectively as in the G-SF algorithm. Use the Hessian inverse to update the probability $p_l(n)$ as follows:
       $$p_l(n+1) = \Gamma_l\left(p_l(n) - b(n)\sum_{k=1}^{K} M_{l,k}(n)Z_k(n)\right) \quad (3.11)$$
5:     Set $n := n + 1$
6: end while
7: Output the routing probability $\boldsymbol{p}(n)$ and terminate.
```

All quasi-Newton algorithms successively approximate the inverse of Hessian $f''\big(x(n)\big)^{-1}$ by the matrix $M(n)$:

$$M(n+1) = M(n) + B(n) \tag{4.1}$$

so that $M(n)$ is positive definite and symmetric for all n. The so called quasi-Newton equation $M(n+1)u(n) = s(n)$ should also be satisfied for all n. The error term $B(n)$ should be minimal. The BFGS update is given by

$$M(n+1) = M(n) - \frac{su^T M(n) + M(n)us^T}{(u,s)} + \left[1 + \frac{(u, M(n)u)}{(u,s)}\right]\frac{ss^T}{(u,s)}, \tag{4.2}$$

see [7] for a derivation and other details.

For the first time we develop quasi-Newton SF algorithms based on the BFGS update. In our problem $x(n)$ is the probability vector $\boldsymbol{p}(n)$ and $f'\big(x(n)\big)$ is the

Algorithm 3. QN-SF algorithm for constrained routing

```
1: Initialize $Z_l(0) = 0$, $u_l(0) = 0$, $p_l(0) = 0.1$, $s_l(0) = 0$ and $M_{i,j}(0) = 0$, $\forall l, i, j = 1, \ldots, K$. Fix $\mathcal{M}$ and $\beta > 0$ and set $n = 0$.
2: while $n < \mathcal{M}$ do
3:     Generate i.i.d $\mathcal{N}(0,1)$ random variables $\boldsymbol{\eta}(n) = \big(\eta_1(n), \eta_2(n), \ldots, \eta_K(n)\big)^T$. Generate two parallel simulations $\{q(n)\}$ and $\{q'(n)\}$ of the queue governed by routing probability parameters $\boldsymbol{p}(n)$ and $\boldsymbol{p}(n) + \beta\boldsymbol{\eta}(n)$.
4:     Use equations (3.4), (3.6), (3.7) and (3.8) for updating $Z_l, Y_i, Y_i'$ and $\lambda_i$, $\forall l = 1, \ldots, K$, $i = 1, \ldots, N$ respectively as in the G-SF algorithm.
5:     Use the BFGS rule (4.2) for approximating the inverse of Hessian matrix $M(n) = [[M_{i,j}(n)]]_{i,j=1}^N$
           $$M(n+1) = M(n) + b(n)\Bigg[ - \frac{\boldsymbol{s}(n)\boldsymbol{u}(n)^T M(n) + M(n)\boldsymbol{u}(n)\boldsymbol{s}(n)^T}{\big(\boldsymbol{u}(n), \boldsymbol{s}(n)\big)} + \left(1 + \frac{\big(\boldsymbol{u}(n), M(n)\boldsymbol{u}(n)\big)}{\big(\boldsymbol{u}(n), \boldsymbol{s}(n)\big)}\right) \frac{\boldsymbol{s}(n)\boldsymbol{s}(n)^T}{\big(\boldsymbol{u}(n), \boldsymbol{s}(n)\big)}\Bigg] \quad (4.3)$$
6:     As in N-SF algorithm use the equation (3.11) for updating the probability $p_l(n+1)$, $l = 1, \ldots K$. Set $n := n+1$ and $\boldsymbol{s}(n) = \boldsymbol{p}(n) - \boldsymbol{p}(n-1)$, $\boldsymbol{u}(n) = \boldsymbol{Z}(n) - \boldsymbol{Z}(n-1)$.
7: end while
8: Output the routing probability $\boldsymbol{p}(n)$ and terminate.
```

vector $\boldsymbol{Z}(n) = [Z_l(n),\ l = 1, \ldots, K]^T$. The vectors $[s_l(n),\ l = 1, \ldots, K]^T$ and $[u_l(n),\ l = 1, \ldots, K]^T$ are denoted by $\boldsymbol{s}(n)$ and $\boldsymbol{u}(n)$ respectively. Note that the BFGS rule for Hessian inverse $M(n)$ and the probability $\boldsymbol{p}(n)$ are updated on the same time scale $b(n)$.

Many line search algorithms like Wolfe's rule or Armijo, Goldstein-Price methods have been used to find the best step-size γ_k given a descent direction. These algorithm calculate the objective function value for many intermediate step sizes. In the network routing problem, the objective function is the delay. This is however not analytically known and it is not feasible to repeat the simulation for small changes in step sizes. Hence line search techniques were not used in SF algorithms.

5 Simulation Results

The algorithms were tested on the setting of multi-stage network of queues, see Fig. 1. All the queues were assumed to be M/M/1 and use FCFS discipline. This assumption is not necessary as the SF algorithms work for any general queue. Packets arrive at the source s and are routed to the destination d by choosing a queue at each stage. The objective is to minimize the long-run average delay for the packets to reach destination d. The constraint that the average queue length at each queue should not exceed a threshold is used.

Individual queue lengths are not used in the algorithms. Expected end-to-end delay is only used for tuning the routing probabilities. This is guided by

Table 1. Expected delay for different λ and μ with first queue ten being times slower

λ/μ	10/5	5/10	5/5	10/10	15/15
Random	2.75±0.003	0.72±0.001	1.51±0.002	1.29±0.001	1.24±0.001
Optimum	0.93±0	0.26±0	0.52±0	0.46±0	0.44±0
G-SF	1.13±0.134	0.33±0.079	0.65±0.049	0.62±0.139	0.57±0.11
QN-SF	0.99±0.182	0.50±0.175	0.67±0.038	0.55±0.032	0.48±0.021
N-SF	1.06±0.259	0.42±0.079	0.61±0.032	0.53±0.107	0.57±0.079

Table 2. Expected delay for different λ and μ with 3 stages and first queue at each stage being ten times slower

λ/μ	10/5	5/10	5/5	10/10	15/15
Random	6.81±0.004	2.88±0.005	5.41±0.011	4.01±0.003	3.47±0
Optimum	1.67±0	0.56±0	1.2±0	0.79±0	0.65±0
G-SF	3.08±0.541	1.57±0.087	2.51±0.412	1.52±0.162	1.76±0.518
QN-SF	3.93±0.356	1.21±0.209	3.71±0.405	1.91±0.4	1.75±0.353
N-SF	3.56±0.553	1.08±0.132	3.76±0.48	2.16±0.163	1.54±0.292

contributions from ACO literature. Given the routing probabilities the expected delay has to be found by simulating the queue. It takes longer time if the network is larger. In our simulations we found the performance of the algorithms was good for a moderately sized network of 3 stages with 4 queues at each stage. The three SF algorithms - Gradient SF, Newton SF and Quasi-Newton SF were tested on this network. The Jacobi version of N-SF using only the diagonal entries in the Hessian matrix was considered. The projection operator P used simply mapped all negative diagonal elements to a small value 0.01.

Table 1 shows the expected delay over 500 iterations for various combinations of total arrival rate λ at the source s and service rate μ_k at the queues. At each iteration we have to simulate the queue to get the steady state values. Here there is only one stage with four queues. The service rate first queue μ_1 is ten times slower than than the rate μ other three queues, $\mu_1 = \mu/10$. This model is useful where there is single router dispatching jobs to N non-identical servers based on the expected job completion time alone. The first row "Random" shows the value for routing with equal probability to all queues. The probabilities are initialized to this value. The second shows the optimum value. The SF algorithms are able to achieve values close to the optimum for this simple case.

Table 2 shows values for the expected delay with three stages and 4 queues at each stage. Hence the parameter dimension for this setting is $16 * 2 + 4 = 36$. Again the first queue in each stage here has ten times slower service rate than the rest. Since it takes longer time to reach the destination, the algorithms performance is lower than optimum but much better than random routing. All the SF algorithms have similar performance for this problem.

Table 3 shows the performance of the algorithms with 4 stages with 4 choices and a constraint that the expected queue length at queue i is lower than $3 * (i \bmod 4 + 1)$. Hence for the first queue in each stage the expected queue length should be lesser than 3, for the second it is 6, third 9 and the last 12.

Table 3. Expected delay for different λ and μ with constraints $\alpha_i = 3*(i \mod 4+1), \forall i$

λ/μ	10/5	5/10	5/5	10/10	15/15
G-SF	1.67±0.015	0.58±0.005	1.21±0.015	0.78±0.014	0.66±0.009
QN-SF	1.65±0.024	0.57±0.006	1.22±0.020	0.78±0.005	0.64±0.011
N-SF	1.65±0.008	0.57±0.026	1.19±0.005	0.78±0.003	0.64±0.006

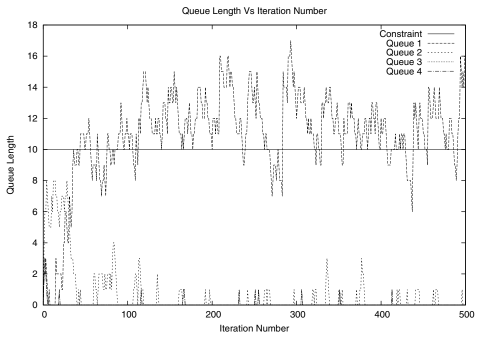


Fig. 2. Queue length Vs Iteration Number. First queue has 10 times higher service rate with no constraint.

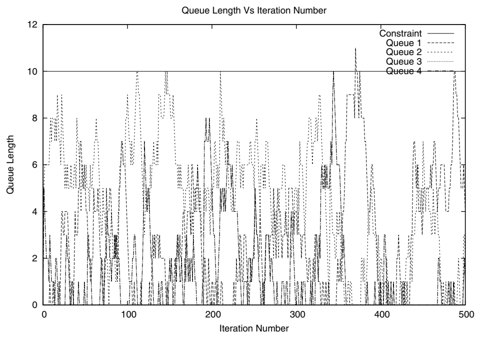


Fig. 3. Queue length Vs Iteration Number. First queue has 10 times higher service rate with constraint on average queue length $\leq$ 10 at all queues.

The plots in Fig. 2 and Fig. 3 show the queue length process for the G-SF algorithm. In this only one stage with four queues was considered. The service rate for the first queue is assumed to be 10 times higher than the rest. In Fig. 2 there are no constraints, hence the queue length at the first queue is much higher than others. While in Fig. 3 the constraint that the expected queue length at all the four queues should not exceed 10 was used. Hence the queue length are more equal at all the queues, though the first one is assumed to be 10 times faster. As expected imposing this constraint results in deterioration in the performance.

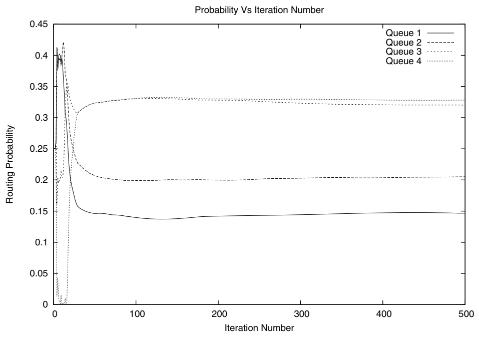


Fig. 4. Routing probability Vs Iteration Number for G-SF algorithm with constraints $\alpha_i = 3 * (i \mod 4 + 1)$, $\forall i$

Fig. 4 shows how the routing probabilities for the four queues at the source vary over time. The constraint is that the expected queue length at queue i is lesser than $3 * (i \mod 4 + 1)$. Initially all the queues have equal probability 0.25. As the G-SF algorithm proceeds the routing probability converges to a value depending on the constraint imposed on the queues. Hence the first queue gets the lowest probability because of the strongest constraint followed by the second, third and the fourth queues. This is clearly seen in the graph. It can also be seen that the difference in probabilities for third and fourth queues is not high. This is because the arrival rate λ is not high enough to violate the weak constraints at these queues.

6 Conclusion

We have developed smoothed functional algorithms for routing in a multi-stage network of queues with constraints on average queue length. For the first time we have proposed a quasi-Newton SF algorithm though variants of this algorithm need to be developed for better performance. Our simulation results show that SF algorithms are able to perform fairly well for this problem where other policies like JSQ,LWL or CQSF cannot be used. Convergence of gradient and Newton SF algorithms is known, however similar proofs need to be shown for the quasi-Newton algorithm. We are also studying reinforcement learning algorithms for a similar problem of routing in a multi-stage queueing network.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Comp. Netw. 38, 393–422 (2002)
2. Bhatnagar, S.: Adaptive newton-based smoothed functional algorithms for simulation optimization. ACM Trans. on Model. Comput. Simul. 18(1), 27–62 (2007)
3. Bhatnagar, S., Hemachandra, N., Mishra, V.: Stochastic approximation algorithms for constrained optimization via simulation (2009) (preprint)

4. Bonnans, J.F., Gilbert, J.C., Lemaréchal, C., Sagastizábal, C.A.: Numerical Optimization: Theoretical and Practical Aspects. Springer, Heidelberg (2006)
5. Borkar, V., Das, D.: A novel aco scheme for emergent optimization via reinforcement and initial bias. Swarm Intell. 3, 3–34 (2009)
6. Cohen, J.E., Kelly, F.P.: A paradox of congestion in a queueing network. J. Appl. Prob. 27, 730–734 (1990)
7. Dennis, J.E., More, J.J.: Quasi-newton methods, motivation and theory. SIAM Review 19(1), 46–89 (1977)
8. Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theor. Comp. Sci. 344, 243–278 (2005)
9. Gupta, V., Harchol-Balter, M., Sigman, K., Whitt, W.: Analysis of join-the-shortest-queue routing for web server farms. Perf. Eval. 64, 1062–1081 (2007)
10. Harchol-Balter, M., Crovella, M., Murta, C.: On choosing a task assignment policy for a distributed server system. J. Parll. Dist. Comput. 59(2), 204–228 (1999)
11. Harrison, P.G., Patel, N.M.: The representation of multistage interconnection networks in queuing models of parallel systems. J. of ACM 37(4), 863–898 (1990)
12. Katkovnik, V.Y., Kulchitsky, Y.: Convergence of a class of random search algorithms. Automat. Remote Contr. 8, 1321–1326 (1972)
13. Kolavali, S., Bhatnagar, S.: Ant colony optimization algorithms for shortest path problems. In: Altman, E., Chaintreau, A. (eds.) NET-COOP 2008. LNCS, vol. 5425, pp. 37–44. Springer, Heidelberg (2009)
14. Spall, J.C.: Adaptive stochastic approximation by the simultaneous peturbation method. IEEE Trans. on Automat. Contr. 45, 1839–1853 (1992)
15. Xiao, X., Ni, L.M.: Internet qos: A big picture. IEEE Netw. 13, 8–18 (1999)

An Incremental Power Greedy Heuristic for Strong Minimum Energy Topology in Wireless Sensor Networks

B.S. Panda and D. Pushpraj Shetty

Computer Science and Application Group
Department of Mathematics
Indian Institute of Technology Delhi, Hauz Khas
New Delhi 110016, India
bspanda@maths.iitd.ac.in, prajshetty@gmail.com

Abstract. Given a set of sensors in the plane, the strong minimum energy topology (SMET) problem is to assign transmit power to each sensor such that the resulting topology containing only bidirectional links is strongly connected. This problem is known to be NP-hard. As this problem is very much significant from application point of view, several heuristic algorithms have been proposed. In this paper, we propose a new incremental power greedy heuristic for SMET problem, called Kruskal-incremental power greedy heuristic. We compare Kruskal-incremental power greedy heuristic with Prim-incremental power greedy heuristic, one of the most popular heuristics available in the literature, through extensive simulation. The simulation results suggest that Kruskal-incremental power greedy heuristic outperforms on an average the Prim-incremental power greedy heuristic.

Keywords: Sensor Networks, Topology Control, Minimum Spanning Tree, Graph Algorithms.

1 Introduction

A wireless sensor network consists of a collection of battery powered sensors each of which is integrated in a single package with low power signal processing, computation, and a wireless transceiver. Many features of wireless sensor networks can be found in an excellent survey paper [2]. Sensor nodes collect data of interest and transmit them to other nodes. In such a network, a packet from a source sensor to the destination sensor may have to travel through intermediate sensors before reaching to the destination. The transmission power of each sensor can be tuned between a minimum and a maximum. The energy requirement by a sensor to transmit a packet depends on its transmission power. Two sensors are said to be connected by a bidirectional link if each of these sensor is in the transmission range of the other. So it is important to adjust the transmit power of each sensors so that the resulting network satisfies some prescribed network properties such as connectivity (see [3, 4, 8, 9]). This problem is known as **topology control problem in sensor networks** and is widely studied (see [5–7, 9, 11, 12]). One of the topology control problems widely studied is the **strong minimum energy topology** problem. The problem is to adjust the transmit power of each sensors so that the resulting network consisting only of the sensors and all bidirectional links is strongly connected, i.e., there is a path between every pair of sensors and the sum of the transmit power assigned to all sensors must be minimum. The **strong minimum energy topology (SMET) problem** is defined as follows: Given a set of sensors in the plane, the strong minimum energy topology (SMET) problem is to assign transmit power to each sensor such that the resulting topology containing only bidirectional links is strongly connected. The SMET problem is shown to NP-hard by Cheng *et al.*[3]. Cheng *et al.* [3] also proposed two heuristics: a minimum spanning tree (MST) based heuristic , and an incremental power greedy. They [3] also showed through simulation results that the incremental power greedy outperforms the MST based heuristic. The incremental power greedy heuristic due to Cheng *et al.* [3] is one of the popular heuristic available in the literature for the SMET problem.

In this paper, a new incremental power greedy heuristic for SMET problem, called Kruskal-incremental power greedy heuristic, is proposed. This heuristic is compared with the MST-heuristic and Prim-incremental power greedy heuristic due to Cheng *et al.* [3] through exhaustive simulation. The Kruskal-incremental power greedy heuristic outperforms the Prim-incremental power greedy heuristic as evident from the simulation results.

The rest of the paper is organized as follows. Section 2 introduces the graph theoretic model of the SMET problem. Section 3 introduces the MST-heuristic and Prim-incremental power greedy heuristic due to Cheng *et al.* [3]. Section 4 presents a new heuristic, called Kruskal-incremental power greedy heuristic, and shows the existence of instances when the proposed heuristic is better than the incremental power greedy heuristic due to Cheng *et al.* [3]. Section 5 presents the simulation results. Finally, Section 6 concludes the paper.

2 A Graph Theoretic Model of the SMET Problem

We present first some graph theoretic terminologies which will be required in this paper. Let $G = (V, E)$ be a graph. Let n and m denote the number of vertices and number of edges of G, respectively. If each edge e of G is assigned a wight $w(e)$, then G is called a weighted graph. A graph $H = (V', E')$ is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. A sequence of vertices $v_1, v_2, \ldots, v_k$ is a path in G if $v_i v_{i+1} \in E$ for all $1 \leq i \leq k-1$. A cycle in G is a sequence of vertices $v_1, v_2, \ldots, v_k, v_1$ such that $v_i v_{i+1} \in E$ for $1 \leq i \leq k-1$ and $v_k v_1 \in E$. A subgraph $T = (V, E')$ of G is called a spanning tree if (i) there is a path in T between every pair of vertices in V, i.e., T is connected, and (ii) T has no cycle, i.e., T is acyclic. The cost of a spanning tree $T = (V, E')$ of a weighted graph $G = (V, E)$ with weight function w is $\sum_{e \in E'} w(e)$. A spanning tree of a weighted graph G is called a minimum spanning tree (MST) if T has the minimum cost among all spanning trees of G. Two popular algorithms for finding an MST of a weighted graph are Prim's algorithm and Kruskal's algorithm (see [1]). Let $T = (V, E')$ be a spanning tree of a weighted graph $G = (V, E)$ having cost function w. Let $P_T(v) = \max\{w(vw)|vw \in E(G)\}$ and $P(T) = \sum_{v \in V} P_T(v)$. Other graph theoretic concepts can be found in [13].

A sensor network consisting of n sensors $s_1, s_2, \ldots, s_n$ in the plane can be modeled as a weighted directed graph G by taking each sensor in the plane as a vertex and joining the vertices s_i and s_j and assigning $w(v_i v_j) = t.d^\alpha$, where d is the Euclidean distance between s_i and s_j, t is a threshold which is a function of signal-to-noise ratio at v_j, and α is a constant that is related to path loss and is from two to four [10]. The SMET problem now reduces to finding a spanning tree T of G such that $P(T)$ is minimum.

3 Summary of Previous Work

The SMET problem is proved to be NP-hard [3]. Chang *et al.* [3] proposed two heuristics for SMET problem. Let $T = (V, E')$ be a subgraph of the complete graph $K_n = (V, E)$ on n vertices. Let $P_T(u)$ be the power assigned to u in T. Note that $P_T(u) = \max\{w(uv)|uv \in E'\}$ if u is adjacent to some vertex v in T; otherwise $P_T(u) = 0$. Let $xy \in E \setminus E'$. Let $T' = (V, E' \cup \{xy\})$. Define $\delta_T(x) = w(xy) - P_T(x)$ if $w(xy) > P_T(x)$ else $\delta_T(x) = 0$. $\delta_T(y)$ is defined similarly. Let $\delta'_T(xy) = \delta_T(x) + \delta_T(y)$. So $\delta'_T(xy)$ is the increase in energy needed to obtain T' from T.

1. **MST-Heuristic:**
   (a) Find a minimum Spanning tree(MST) T of G. Let $P(u)$ be the power assigned to u for all $u \in V$ and P be the sum of all $P(u)$.
   (b) Compute $P(u) = \max\{w(uv)|uv \in E(T)\}$.
   (c) Output P and $P(u)$ for all $u \in V$.
2. **Prim-incremental power greedy Heuristic:**
   (a) **Initialization:** Let S be the set containing the subset of sensors considered so far during the execution of the heuristic. Let $T = (S, E')$. Let P be the total power of all the sensors in S, and $P(u)$ be the power expenditure in sensor u. Initially $P = 0, S = \{v_0\}$, where v_0 is any sensor $,P(v_0) = 0$, and $E' = \emptyset$.
   (b) Let $S\prime = V - S$. Find $u \in S$ and $v \in S\prime$ such that $\delta'_T(uv)$ is minimum among all $u \in S$ and $v \in V \setminus S$, i.e., connecting u and v needs minimum incremental power $\delta'_T(uv)$. Set $S = S \cup \{v\}, P = P + \delta'_T uv$.
   (c) If $S = V$, output P and $P(v)$ for all $v \in V$, and stop; else goto step (b).

The Prim-incremental power greedy heuristics builds the tree more like the Prim's MST Algorithm. Hence we call it Prim-incremental power greedy heuristic.

One of the ways to construct an MST of a weighted graph is to use Prim's MST algorithm [1]. If Prim's algorithm is used to construct the MST in the

MST-heuristic, then in each iteration an edge uv is added to T if $\delta_T(v) = w(vu)$ is minimum. However, even though $\delta_T(v)$ is minimum, to reach v, u may need additional energy $\delta_T(u)$ and in turn $= \delta'_T(uv) = \delta_T(u) + \delta_T(v)$ may not be minimum. This has been taken care of in the Prim-incremental power greedy heuristic.

4 Kruskal-Incremental Power Greedy Heuristic

It has been shown in [3] that MST-Heuristic is a 2-approximation algorithm, i.e., the total power P needed by MST-Heuristic is at most twice the optimal power. Though no performance guarantee for Prim-incremental power greedy heuristic has been obtained, it has been shown by Cheng *et al.* [3] through extensive simulation that Prim-incremental power greedy heuristic outperforms MST-heuristic. However, Prim-incremental power greedy heuristic suffers from the following demerits.

1. The performance of the algorithm heavily dependents on the initial vertex chosen.
2. It finds local minimum in each stage while selecting an edge with minimum incremental power.

As we need to construct a spanning tree with minimum power, we need to choose $n-1$ edges so that these edges do not for any cycle and the total power required is as small as possible. The demerits of Prim-incremental power greedy heuristic is that it enforces that the so far selected edges forms a connected component in addition to the constraint that these edges do not form any cycle. The connected component constraint forces the Prim-incremental power greedy heuristic to select an edge based on local minimum. The performance of Prim-incremental power greedy heuristic also heavily depends upon the initial vertex chosen. These two demerits can be fixed by removing the constraint that the so far selected edges forms a single connected component and the initial vertex is the one which needs minimum incremental power to connect to another node. This is the motivation of our new heuristic.

We next describe a new incremental power greedy heuristic which is based on Kruskal's MST algorithm and hence we call it Kruskal-incremental power greedy heuristic.

In the Kruskal-incremental power greedy heuristic, we select an edge xy such that the edge xy needs minimum incremental energy among all edges which are not selected so far subject to the condition that it does not form a cycle with the so far selected edges. The heuristic is described below.

Kruskal-incremental Power Greedy Heuristic:

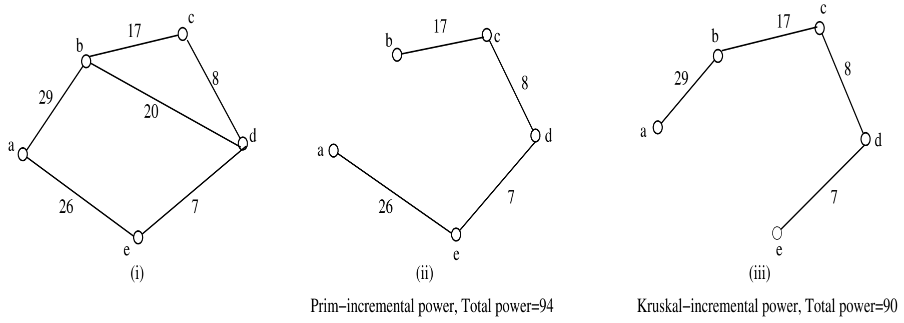
1. Initialization: $T = (V, E')$, where $E' = \emptyset$, $P = 0, P_T(u) = 0$ for all $u \in V$.
2. Find an edge $xy \in E \setminus E'$ such that (i) $T = (V, E' \cup \{xy\})$ is acyclic and (ii) $\delta'_T(xy)$ is minimum.

3. $E' = E' \cup \{xy\}, P_T(x) = P_T(x) + \delta_T(x), P_T(y) = P_T(y) + \delta_T(y),\ P = P + \delta'_T(xy)$.
4. if $|E'| = n - 1$, then output $T = (V, E')$, P, and $P_T(v)$ for all $v \in V$ and stop; else go to step 2.

We now analyze the time complexity of Kruskal-incremental power greedy heuristic.

Lemma 1. *The Kruskal-incremental power greedy heuristic takes $O(n^2 \log n)$ time, where n is the number of nodes in the network.*

Proof. Let E_{i-1} be set set of edges that have been selected till $(i-1)th$ iteration. In ith iteration the algorithm chooses an edge xy such that $E_{i-1} \cup \{xy\}$ is acyclic and $\delta'_T(xy)$ is minimum, where $T = (V, E_{i-1})$. Using the disjoint set data structure, acyclicity testing in all the $n-1$ iterations can be done in $O(m \log m)$ time [1]. The minimum finding takes $O(\log m)$ time in each iteration if we use augmented minimum heap of the $\delta'_T(xy)$ of all the edges not in E_{i-1}. Once an edge xy is added to E_{i-1} to get E_i, we need to update $\delta'_T(ab)$ of all edges in $E \setminus E_i$ which are incident either on x or on y. So $O(n)$ updates are needed per iteration. Since each update in an augmented heap can be done in $O(\log m)$ time [1], all updates take $O(n^2 \log m) = O(n^2 \log n)$ time. Hence Kruskal-incremental power greedy heuristic takes $O(n^2 \log n)$ time. □



c
b
17
29
20
8
a
d
26
7
e
(i)
b
17
c
8
d
a
26
7
e
(ii)
Prim–incremental power, Total power=94
c
b
17
29
8
a
d
7
e
(iii)
Kruskal–incremental power, Total power=90

Fig. 1. Prim's Vs. Kruskal incremental power greedy(Kruskal is better)

Figure 1 illustrates Prim-incremental power greedy heuristic and Kruskal-incremental power greedy heuristic algorithms. The cost of each missing edges is 100 and can be seen easily that none of the missing edges will be selected by each of these two heuristics. Hence these high cost edges are not shown in the graph G of Figure 1(i). The tree constructed by Prim-incremental power greedy heuristic is shown in Figure 1(ii) Initially, $T = (S, E')$, where $S = \{a\}$ and $E' = \emptyset$. As the stating vertex is a, we have two choices, namely ae and ab. However, $\delta'_T(ae) = 52$ and $\delta'_T(ab) = 58$. So, ae is chosen. and $E' = \{ae\}$. In next stage, we have two choices, namely ab and ed. Now $\delta_T(a) = 3$, $\delta_T(b) = 29$, $\delta_T(e) = 0$, and $\delta_T(d) = 7$. So, $\delta'_T(ab) = 3 + 29 = 32$ and $\delta'_T(ed) = 0 + 7 = 7$. So, ed is chosen next. Similarly, it can be seen that Prim-incremental power greedy heuristic chooses the edges ae, ed, dc and ab in this order. The power needed by Prim-incremental power greedy heuristic is 94. However, Kruskal-incremental power greedy heuristic chooses the edges ed, dc, bc, and ab in this order and produces the tree T_2 with power 90 as shown in Figure 1(iii). So Kruskal-incremental power greedy heuristic requires 4 unit less power than the power required by Prim-incremental power greedy heuristic in this case.

The following lemma shows that the difference in the power required by Kruskal-incremental power greedy heuristic and Prim-incremental power greedy heuristic can be arbitrarily large.

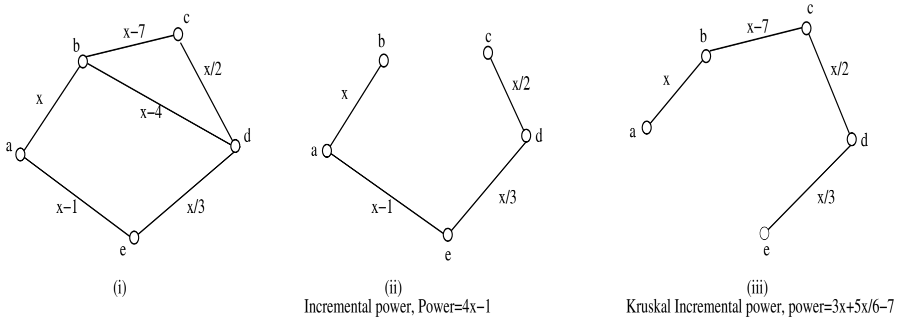

b
c
x−7
x/2
x
x−4
a
d
x−1
x/3
e
(i)
b
c
x
x/2
a
d
x−1
x/3
e
(ii)
Incremental power, Power=4x−1
c
b
x−7
x
x/2
a
d
x/3
e
(iii)
Kruskal Incremental power, power=3x+5x/6−7

Fig. 2. Power difference between Prims and Kruskal-incremental can be large

Lemma 2. *Given any integer $k > 6$, there is a weighted complete graph with five vertices such that the Prim-incremental power greedy heuristic takes k more units of power than the power required by Kruskal-incremental power greedy heuristic.*

Proof. Consider the weighted complete graph G with weight system as shown in Figure 2(i). The costs of the edges which are not shown in the graph are very high so that these edges will not be included by any of the two heuristics considered in the lemma. One way to achieve this is by setting the cost of each missing edge to be x^2. The Prim-incremental power greedy heuristic when applied to G of Figure 2(i), produces the tree T shown in Figure 2(ii). The Kruskal-incremental power greedy heuristic produces the tree T shown in Figure 2(iii). Kruskal-incremental power greedy heuristic chooses the edges ed, dc, bc and ab in this order. The total power needed by Prim-incremental power greedy heuristic is $4x - 1$ whereas the total power needed by Kruskal-incremental power greedy heuristic is $3x + 5x/6 - 7$. So the difference in power is $x/6 + 6$. If we choose $x = 6k - 36$, then Prim-incremental power greedy heuristic will need k more unit of power needed by Kruskal-incremental power greedy heuristic for the graph G of Figure2(i). This proves the lemma. □

5 Experimental Results

We compare all the three heuristics, namely, (*i*) MST-heuristic, (*ii*) Prim-incremental power greedy heuristic, and (*iii*) Kruskal-incremental power greedy heuristic. We assume n sensors are randomly distributed in a 1000×1000 square. The power function used in the simulation study is $f(d) = t.d^{\alpha}$, where α is a constant between 2 and 4. We take $\alpha = 2$ in our simulation study, t is the threshold which is set to 1. For each n ranging from 10 to 100 in increments of 5, we run the heuristics 100 times with different seeds for random number generator. The average of the total powers is reported in Figure 3. The maximum and the variance of power consumptions are shown in Figure 4 and 5 respectively. Table 1 shows the number of times Kruskal-incremental power greedy heuristic outperforms Prim-incremental power greedy heuristic out of 100 runs. Since we plot the average of 100 runs for each n considered in the paper, the comparison of variance of power consumption is essential to see the stability of the heuristic.

We find that total energy decreases with the increase in number of nodes. This is because when the sensors are densely located, it requires less energy to reach the neighbor. We also observe that 66 percent of the time Kruskal-incremental power greedy heuristic requires less total power than the Prim-incremental power greedy heuristic. Total transmit power produced by Kruskal-incremental power greedy heuristic is 0.67 percent less than that of Prim-incremental power greedy

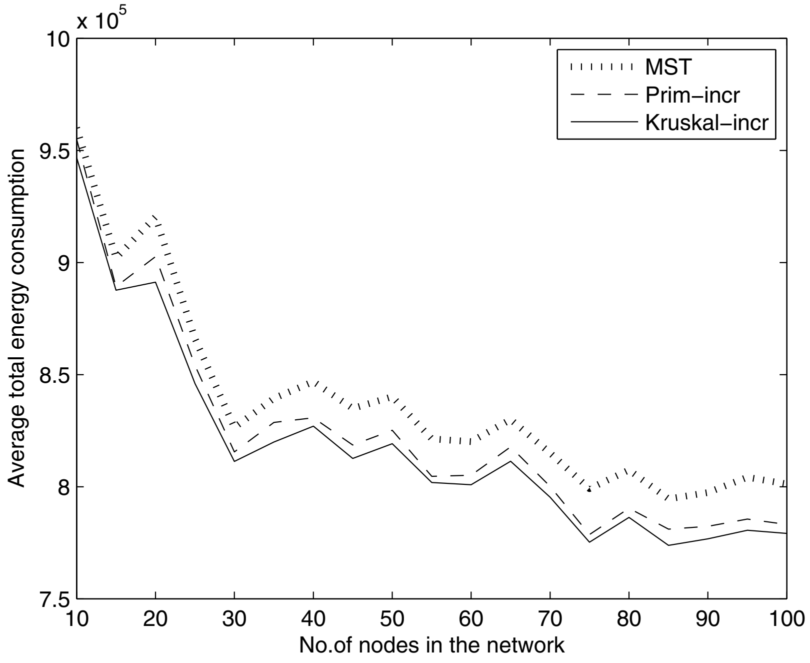


Fig. 3. Average total power

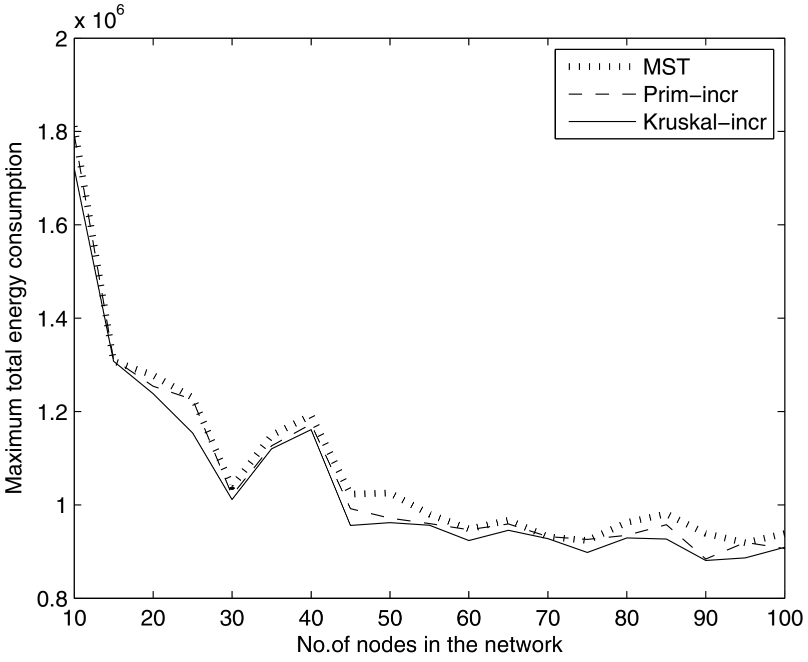


Fig. 4. Maximum total power

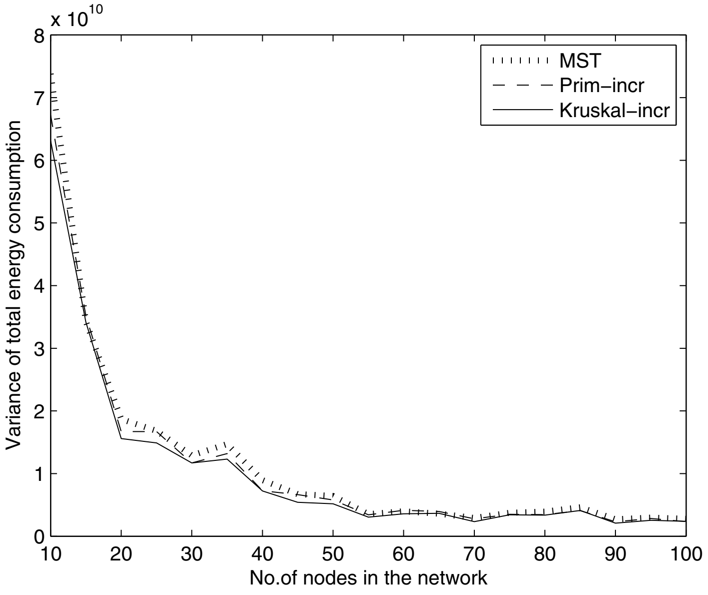


Fig. 5. Variance of total power

Table 1. Comparison of Prim-incremental power greedy and Kruskal-incremental power greedy Heuristics

No.of Nodes	MST	Prim-incr	Kruskal-incr	%Avg Imp	%better
10	960130	954950	946788	0.85	45
15	902063	889198	887646	0.17	74
20	920350	902596	891203	1.26	68
25	865974	853997	846003	0.94	69
30	826402	815477	811237	0.52	55
35	839361	828701	819911	1.06	72
40	846873	830740	827007	0.45	60
45	834892	818632	812645	0.73	55
50	840166	825165	819163	0.73	70
55	821330	804650	801864	0.35	65
60	820065	805089	800832	0.53	63
65	829995	817755	811348	0.78	65
70	815051	800098	795336	0.6	66
75	799142	778653	775288	0.43	62
80	807427	790382	786310	0.52	63
85	794559	781058	773804	0.93	76
90	797419	782288	776716	0.71	63
95	804207	785587	780546	0.64	59
100	801213	783156	779163	0.51	62

heuristic. The maximum transmit power produced by Kruskal-incremental power greedy heuristic is 1.78 percent less than that of Prim-incremental power greedy heuristic. The variance of total power consumption is improved by 6.87 percent in compared to Prim-incremental power greedy heuristic. This study indicates that Kruskal-incremental power greedy heuristic performs better than the Prim-incremental power greedy heuristic.

6 Conclusion

In this paper, we proposed a new heuristic, called Kruskal-incremental power greedy, for strong minimum energy topology problem in wireless sensor networks. We compared the proposed heuristic with Prim-incremental power greedy heuristic and MST-heuristic proposed in [3]. Our study shows that the proposed heuristic out performs both MST-heuristic as well as Prim-incremental power greedy heuristic. However, there are cases in which Kruskal-incremental power greedy is worse than that of Prim-incremental power greedy heuristic, as can be seen from Table 1. So there is a scope of improving the Kruskal-incremental power greedy heuristic using some novel ideas so that in each case it is better than Prim-incremental power greedy heuristic. Note that MST-heuristic is a 2-approximation algorithm. As both Prim-incremental power greedy heuristic and Kruskal-incremental power greedy heuristic are better than MST-heuristic, it would be interesting to see whether these heuristics are 2-approximate algorithms.

References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: Data Structures and Algorithms. Addison-Wesley Publishing Company, Reading (1987)
2. Akyildiz, I.F., Su, W., Sankarsubramanian, Y., Cayirci, E.: Wireless Sensor Networks: A Survey. Computer Networks 38, 393–422 (2002)
3. Cheng, X., Narahari, B., Simha, R., Cheng, M., Liu, D.: Strong minimum energy topology in wireless sensor networks: NP-Completeness and Heuristics. IEEE Transactions on Mobile Computing 2(3), 248–256 (2003)
4. Cheng, M.X., Cardei, M., Sun, J., Cheng, X., Wang, L., Xu, Y., Du, D.-Z.: Topology Control of Ad Hoc Wireless Networks for Energy Efficiency. IEEE Transactions on Computers 53(12), 1629–1635 (2004)
5. Gonzales, T. (ed.): Handbook of Approximation Algorithms and Metaheuristics, ch. 67. Chapman and Hall CRC, Boca Raton (2007)
6. Kirousis, L.M., Kranakis, E., Krizane, D., Pele, A.: Power consumption in packet radio networks. Theoretical Computer Science 243, 289–305 (2000)
7. Labrador, M.A., Wightman, P.M.: Topology Control in Wireless Sensor Networks. Springer, Heidelberg (2009)
8. Li, D., Du, H., Liu, L., Huang, S.C.H.: Joint Topology Control and Power Conservation for Wireless Sensor Networks Using Transmit Power Adjustment. In: Hu, X., Wang, J. (eds.) COCOON 2008. LNCS, vol. 5092, pp. 541–550. Springer, Heidelberg (2008)
9. Lloyd, E.L., Liu, R., Marathe, M.V., Ramanathan, R., Ravi, S.S.: Algorithmic aspects of topology control problems for Ad Hoc Networks. Mobile Networks and Applications 10, 19–34 (2005)
10. Rappaport, T.S.: Wireless communications: Principle and Practice. Prentice-Hall, Englewood Cliffs (1996)
11. Santi, P.: Topology Control in wireless ad hoc and sensor Networks. ACM Computing Surveys 37(2), 164–194 (2005)
12. Santi, P.: Topology Control in wireless ad hoc and sensor Networks. Wiley Inter Science, Chichester (2005)
13. West, D.: Introduction to Graph Theory. PHI (2006)

k^{th} Order Geometric Spanners for Wireless Ad Hoc Networks

Prabhat Kiran and S.V. Rao

Department of Computer Science & Engineering,
Indian Institute of Technology
Guwahati 781039, Assam, India
{p.kiran,svrao}@iitg.ernet.in

Abstract. Wireless ad hoc network can be modeled as a *unit disk graph* (UDG) in which there is an edge between two nodes if and only if their Euclidean distance is at most one unit. The size of UDG is in $O(n^2)$, where n is the number of network nodes. In the literature, the geometric spanners like Relative Neighborhood Graph (RNG), Gabriel Graph (GG), Delaunay Triangulation (Del), Planarized Localized Delaunay Triangulation (PLDel) and Yao Graph are proposed, which are sparse subgraphs of UDG. In this paper, we propose a hierarchy of geometric spanners called the k^{th} order RNG (k-RNG), k^{th} order GG (k-GG), k^{th} order Del (k-Del), and k^{th} order Yao (k-Yao) to reduce the spanning ratio and control topology, sparseness and connectivity. We have simulated these spanners and compared with the existing spanners. The simulation results show that the proposed spanners have better properties in terms of spanning ratio and connectivity by controlling topology and sparseness.

Keywords: Geometric Spanners, Wireless Networks, Distributed Computing.

1 Introduction

Wireless ad hoc network can be modeled as a unit disk graph, UDG(V), in which there is an edge between two nodes if and only if their Euclidean distance is at most one unit, where V denotes the set of nodes in the network. Routing and topological control often require subgraphs of UDG(V), which are sparse, can be constructed locally in an efficient way, and is still relatively good compared with the original unit disk graph for routes' quality. One such quality requirement is the shortest path connecting any two nodes in the subgraph is not much longer than the shortest path connecting them in the original unit disk graph. This aspect of path quality is called the *stretch factor* of the subgraph. A subgraph with a constant stretch factor is often called a *spanner*.

A geometric graph G is a t-spanner (for $t \geq 1$) when the length of the shortest path in G between any pair of nodes a, b does not exceed $t \cdot |ab|$ where $|ab|$ is the Euclidean distance between a and b. Any path from a to b in G whose length

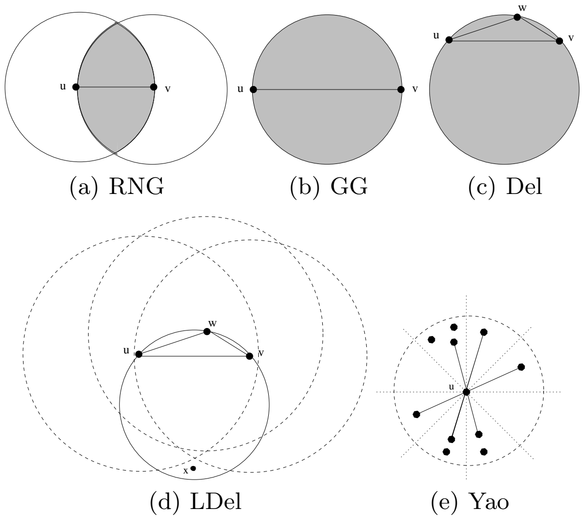


Fig. 1. Neighborhood properties used by various geometry spanners

does not exceed $t \cdot |ab|$ is a t-spanning path. The smallest constant t having this property is the *spanning ratio* or *stretch factor* of the graph. A comprehensive survey on the topic can be found in [1].

Several geometric structures are known to be spanners and can be exploited for use in ad hoc wireless networks [2]. Here we review some of the spanners related to our work. Let UDG of V be denoted by G.

The *relative neighborhood graph* (RNG(V)), consists of all edges uv such that there is no point $w \in V$ such that $|uw| < |uv|$ and $|vw| < |uv|$ [3]. Thus an edge uv is included if the intersection of two circles centered at u and v with radius $|uv|$ does not contain any vertex w from the set V (see the Fig. 1(a)). The *RNG* is a planar graph. Its length stretch factor, however, is at most $n - 1$.

The *Gabriel graph* (GG(V)), consists of all edges uv such that $disk(u, v)$ does not contain any node from V, where $disk(u, v)$ is the disk with diameter $|uv|$ (see the Fig. 1(b)). The *GG* is a planar graph. The length stretch factor of $GG(V)$ is at most $\frac{4\pi\sqrt{2n-4}}{3}$ [4].

The emptiness criteria of RNG and GG is tested by each node with respect to its 1-hop neighbors, instead of V, in localized distributed construction for adhoc networks.

A triangulation of V is a *Delaunay triangulation* (Del(V)), if the circumcircle of each of its triangles does not contain any other vertices of V in its interior (see the Fig. 1(c)). The Del(V) is a planar t-spanner with spanning ratio $\frac{2\pi}{3\cos\frac{\pi}{6}} = \frac{4\sqrt{3}}{9}\pi \approx 2.42$ [5] [6]. The best known lower bound on t is $\pi/2$ [7]. The emptiness criteria of circumcircle of a triangle is tested with respect to 1-hop neighbors of each node of the triangle in localized distributed construction.

A localized algorithm that constructs a sequence of graphs, called *localized Delaunay* ($LDel^m(V)$) is proposed by Li et al [8]. An edge uv is called a *Gabriel edge* if $|uv| \leq 1$ and the open disk with uv as diameter does not contain any vertex from V. We call a triangle $\triangle uvw$ a *m-localized Delaunay triangle* if the interior of the circumcircle of $\triangle uvw$, does not contain any vertex of V that is a m-neighbor of u, v, or w (see the Fig. 1(d)) and all edges of the triangle $\triangle uvw$ have length no more than one unit. Then, the *m-localized Delaunay graph* over a vertex set V has exactly all unit Gabriel edges and edges of all m-localized Delaunay triangles. A node q is a m-neighbor of p if q is in m-hop neighborhood of p. The $LDel^m(V)$ is a planar graph for any $m \geq 2$ [8]. A planar graph $PLDel(V)$ is constructed from $LDel^{(1)}(V)$ locally and is shown to be a t-spanner of $UDG(V)$.

The *Yao graph* with an integer parameter $c \geq 6$, $\overrightarrow{YG}_c(V)$, is defined as follows. At each node u, c cones are defined by taking c equally separated rays originated at u. In each cone, choose the closest node v to u with distance at most one, if there is any, and add a directed link $\overrightarrow{uv}$, ties are broken arbitrarily (see the Fig. 1(e)). Let $YG_c(V)$ be the undirected graph by ignoring the direction of each link in $\overrightarrow{YG}_c(V)$. Yao graph contains at most cn edges. Its length stretch factor is at most $\frac{1}{1-2\sin\frac{\pi}{c}}$. However, the Yao graph is not guaranteed to be planar. The directed graphs $\overrightarrow{YG}_c(V)$ has a bounded out-degree c, but some nodes can have large in-degree.

The existing spanners provide fixed topologies, which does not allow to change spanning ratio, sparseness and connectivity. In this paper we proposed higher order graphs based on the existing spanners, for adhoc networks, to reduce the spanning ratio and control topology, sparseness and connectivity. We have simulated these spanners and compared with the existing spanners. The simulation results show that the proposed spanners have better properties in terms of spanning ratio and connectivity by controlling topology and sparseness.

The rest of the paper is organized as follows. In the next section, we define our proposed graphs and their properties. The third section presents a detailed analysis of simulation results. The last and final section concludes with some pointers to future research directions.

2 k^{th} Order Geometric Spanners

In the literature, higher order geometric graphs, k-GG, k-RNG, and k-Del are used to solve the problems Hamiltonian cycle [9], and Euclidean bottleneck matching [10], and Euclidean biconnected edge subgraph [11]. The k-GG and k-RNG can be computed in $O(kn^{3/2}\log n + k^2 n)$ time [12]. This algorithm is a centralized algorithm which is not suitable for ad-hoc networks. More over each node has to know the position of all the nodes in the network, which is very expensive. The criteria of higher order graph edges are tested with respect to its 1-hop neighbors, instead of V, in localized distributed construction for adhoc networks.

2.1 k^{th} Order Relative Neighborhood Graph (k-RNG)

The k^{th} *order Relative Neighborhood Graph* consists of all edges uv such that there are no more than $k-1$ nodes $w \in V$ such that $|uw| < |uv|$ and $|wv| < |uv|$. In other words the edge uv is included in the graph if and only if the intersection of two circles centered at u and v and with radius $|uv|$, denoted by $lune(u, v)$, does not contain more than $k-1$ nodes w from the set V. Note that for $k = 1$, *k-RNG* reduces to the normal *RNG*.

In order to make the graph construction amenable to ad-hoc networks, we relax the criteria of checking $lune(u, v)$ restricted to 1-hop neighbors instead of V. In other words. each node u places an edge to a node v if and only if $lune(u, v)$ does not contain more than $k-1$ nodes of 1-hop neighbors of u. This method is depicted in the algorithm below.

```
Algorithm: $k$-RNG

Broadcast hello_packet with ID and location information
    and collect 1-hop information $n_1(u)$.
Each node $u$ follow the steps to find $k$-RNG edges
    for each $a_i \in n_1(u)$ do
        if $lune(u, a_i)$ contain less than $k$ nodes $a_j \in n_1(u)$
            add edge $ua_i$ to $k$-RNG.
```

We can prove the following interesting properties of k-RNG.

Lemma 1. *k-RNG $\subseteq$ $(k+1)$-RNG.*

Lemma 2. *The message complexity of construction of k-RNG is in $O(n)$, where n is number of nodes in the network.*

Lemma 3. *The time complexity of construction of k-RNG is in $O(\Delta^2)$, where Δ is the maximum node degree of UDG.*

Proof. It follows from the fact that the time required to check $lune(u, a_i)$ contains at most $k-1$ nodes is in $O(\Delta)$, since number of nodes in 1-hop neighbors for the node u is in $O(\Delta)$. This test has to done for each edge formed between u and every 1-hop neighbor of u, which are at most Δ. Hence follows the lemma. ∎

2.2 k^{th} Order Gabriel Graph (k-GG)

The k^{th} order Gabriel Graph (k-GG) contain all edges uv such that the $disk(u, v)$ does not contain more than $k-1$ nodes from V. *k-GG* reduces to the normal *GG*, for $k = 1$. Similar to k-RNG, we relax the criteria of checking the $disk(u, v)$

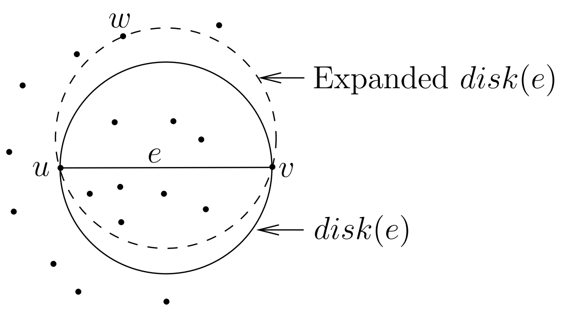


Fig. 2. Illustration of the proof k-GG $\subseteq$ k-Del

to 1-hop neighbors only. That is, each node u places an edge to the node v if and only if $disk(u, v)$ does not contain more $k - 1$ nodes of 1-hop neighbors of u. One can easily verify the following lemmas.

Lemma 4. *k-GG $\subseteq$ $(k + 1)$-GG.*

Lemma 5. *The time and message complexity of construction of k-GG is in $O(\Delta^2)$ and $O(n)$ respectively, where Δ is the maximum node degree of UDG and n is number of nodes.*

Lemma 6. *k-RNG $\subseteq$ k-GG, for given k.*

Proof. Let e be an edge in k-RNG. This implies that the $lune(e)$ contains at most $k - 1$ nodes. The edge $e \in k$-GG, because $disk(e) \subsetneq lune(e)$. ∎

2.3 k^{th} Order Delaunay Graph (k-Del)

The k^{th} *order Delaunay Graph*, consists of all edges uv such that the interior of the circumcircle of the triangle $\triangle uvw$ does not contain more than $k - 1$ nodes from the set V. For $k = 1$, *k-Del* reduces to the normal *Del*. Similar to previous graphs, we relax the criteria of checking the circumcircle of $\triangle uvw$ to 1-hop neighbors of u only. Some properties of the graph are stated below.

Lemma 7. *k-Del $\subseteq$ $(k + 1)$-Del.*

Lemma 8. *The time and message complexity of construction of k-Del is in $O(\Delta^3)$ and $O(n)$, where Δ is the maximum node degree of the network in UDG and n is number of nodes.*

Lemma 9. *k-GG $\subseteq$ k-Del, for given k.*

Proof. Let e be an edge in k-GG. This implies that there exits a $disk(e)$ which contain at most $k - 1$ nodes as shown in the Fig. 2. Expand the $disk(e)$ till it touches a node w outside of it such that the end points of e, u and v, also lies on the circle as shown in the Fig. 2. The expanded circle is the circumcircle of $\triangle uvw$ and contains at most $k-1$ nodes. ∎

Hence follows the theorem.

Theorem 1. *k-RNG $\subseteq$ k-GG $\subseteq$ k-Del.*

2.4 k^{th} Order Yao Graph (k-Yao)

The k^{th} *order Yao Graph* is formed by considering each node $u \in V$ and connecting it to no more than k closest nodes v in each cone, and preserving only those edges with $|uv| \leq 1$. The number of cones is given by a parameter c, which is the number of equally separated angular regions around any given node. For $k = 1$, k-Yao reduces to the normal Yao graph.

Lemma 10. *The time and message complexity of construction of k-Yao is in $O(\Delta)$ and $O(n)$, where Δ is the maximum node degree of the network in UDG and n is the number of nodes.*

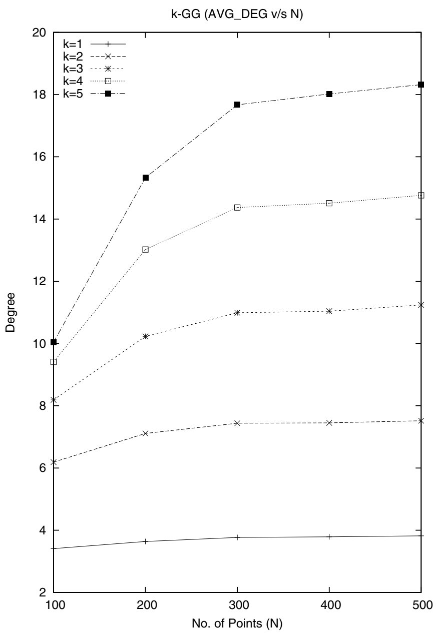
3 Simulation and Analysis

We have simulated these higher order geometric graphs and evaluated their performance. We have varied the number of nodes between 100 and 500 in the area of $500 \times 500m^2$ with each nodes' communication radius of $100m$.

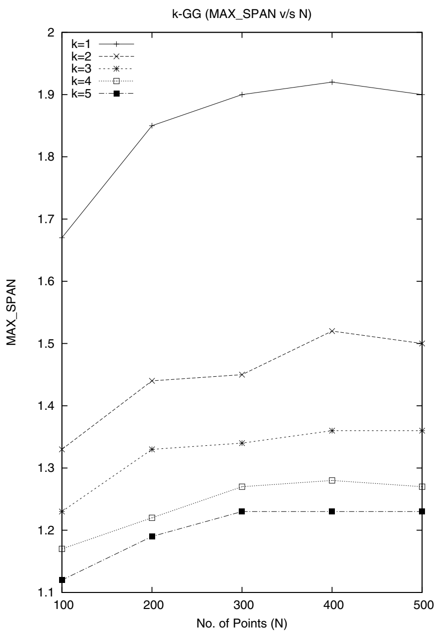
We have evaluated our proposed structures against average degree of nodes (AVG_DEG), maximum spanning ratio (MAX_SPAN), percentage of UDG edges present in the graph (%UDG), and average number of edges per node (SIZE_FAC) by varying the number of nodes in the network between 100 and 500. Similar to graph size, SIZE_FAC gives an estimate of sparseness of the graph. The other properties like minimum degree, average spanning ratio, graph size and diameter are studied in [13].

The simulation results are shown in the Fig. 3, Fig. 4, Fig. 5, Fig. 6, and Fig. 7. For a given k, as number of nodes increases, the graph size naturally increases. But the %UDG decreases, because large number of edges do not satisfy the k^{th} order graph property due to increase in node density. However, spanning ratio increases since edges density increased in UDG and decreased in higher order graphs. This means that the sparseness of the graph with respect to UDG is increasing as N increases.

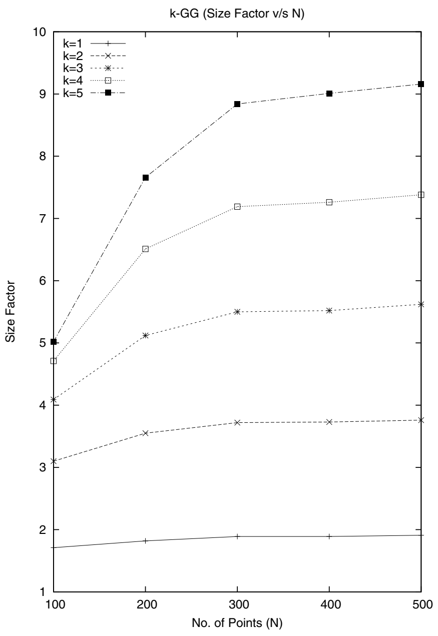
As k increases, connectivity between nodes increase due to relaxation of the higher order graph property, which can observed in increase in average node degree, average number of edges per node and percentage of UDG edges. This leads to decease in spanning ratio and sparseness. Moreover, increase in the connectivity increases the fault tolerance of network.



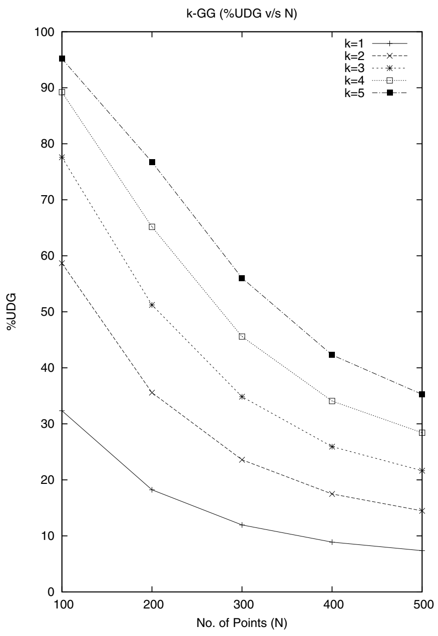
(a) Average Degree v/s Points



(b) Maximum Spanning Ratio v/s Points

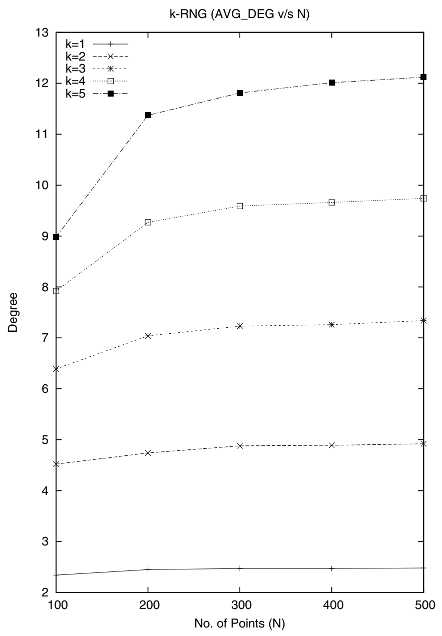


(c) Size Factor v/s Points

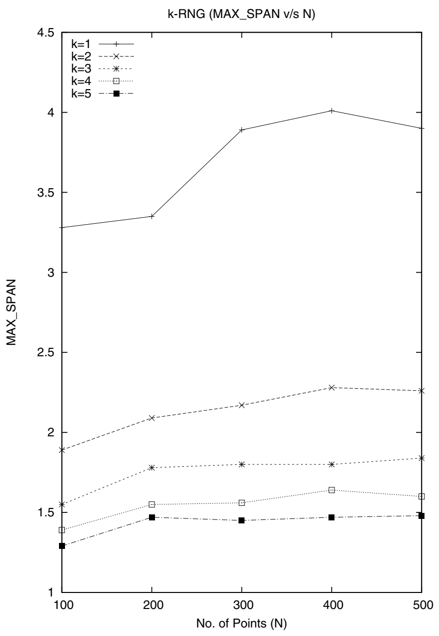


(d) Percentage of UDG v/s Points

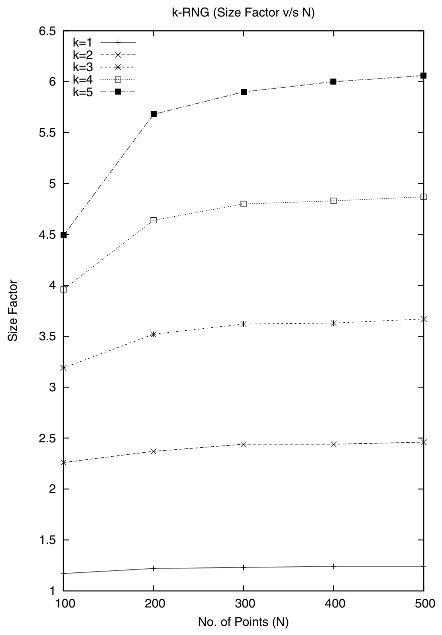
Fig. 3. Simulation results for k-GG



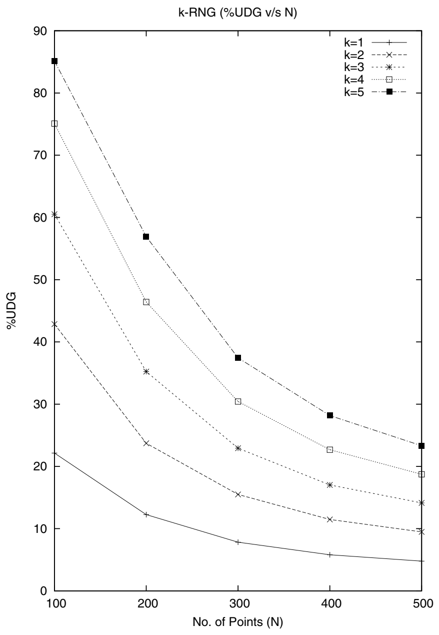
(a) Average Degree v/s Points



(b) Maximum Spanning Ratio v/s Points

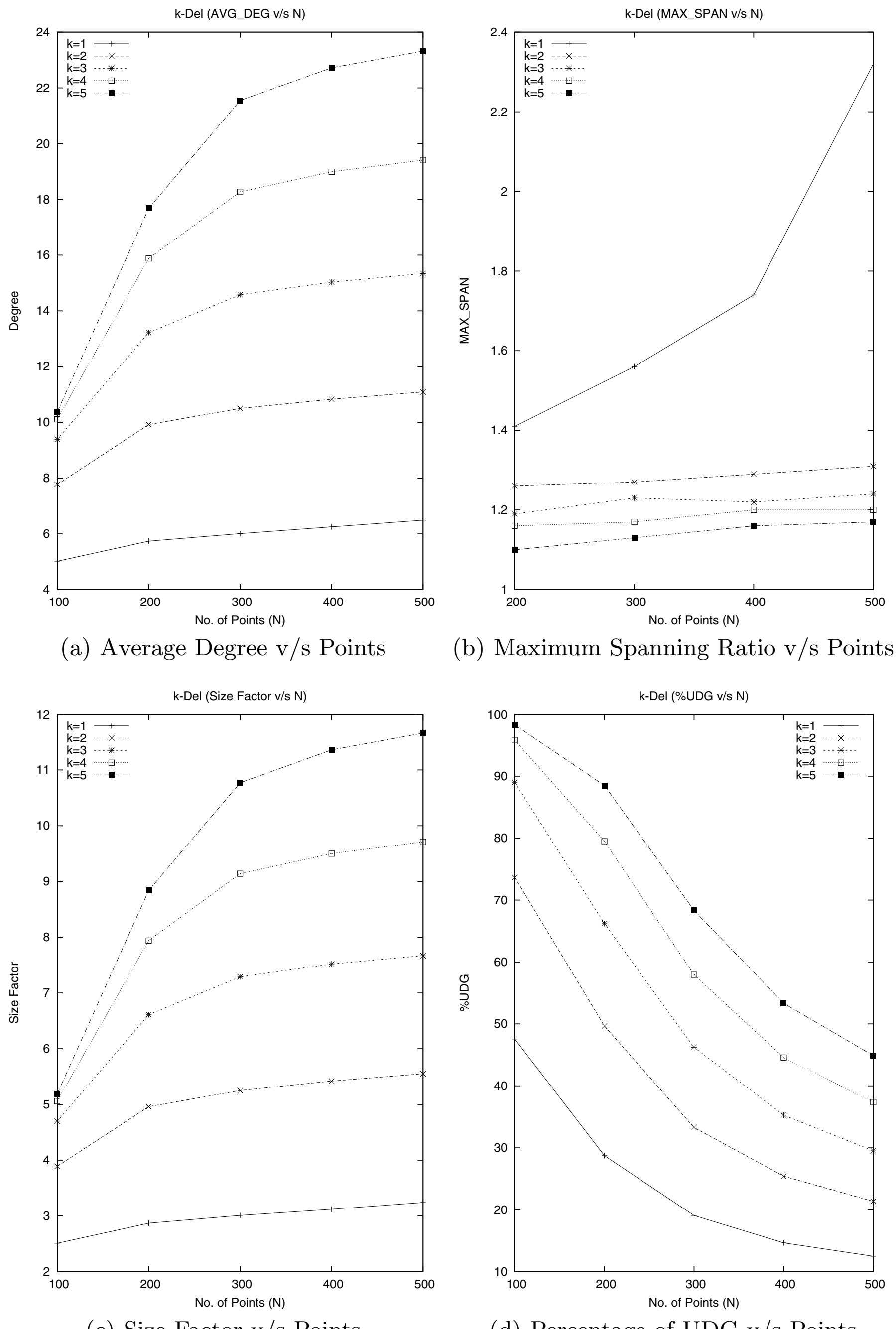


(c) Size Factor v/s Points



(d) Percentage of UDG v/s Points

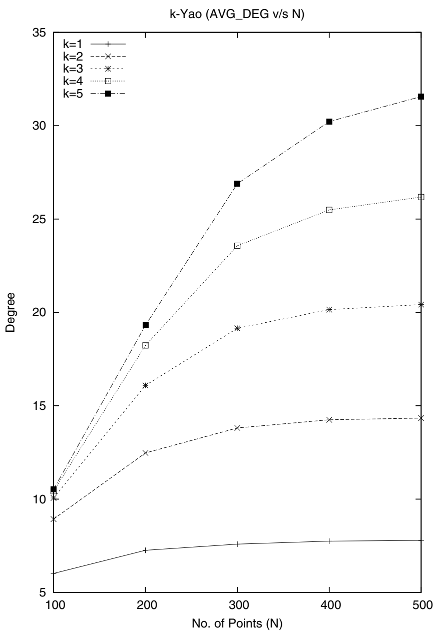
Fig. 4. Simulation results for *k*-RNG



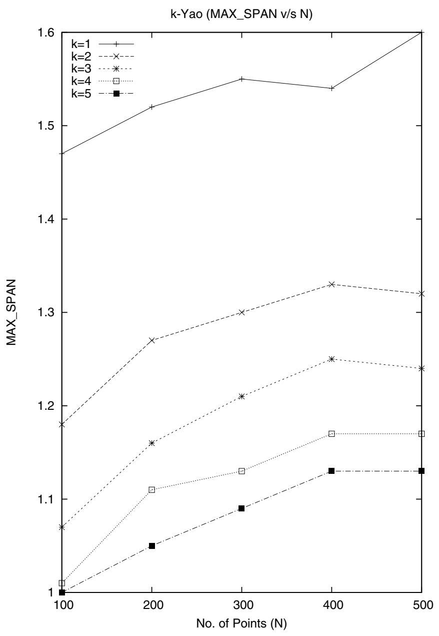
(a) Average Degree v/s Points (b) Maximum Spanning Ratio v/s Points

(c) Size Factor v/s Points (d) Percentage of UDG v/s Points

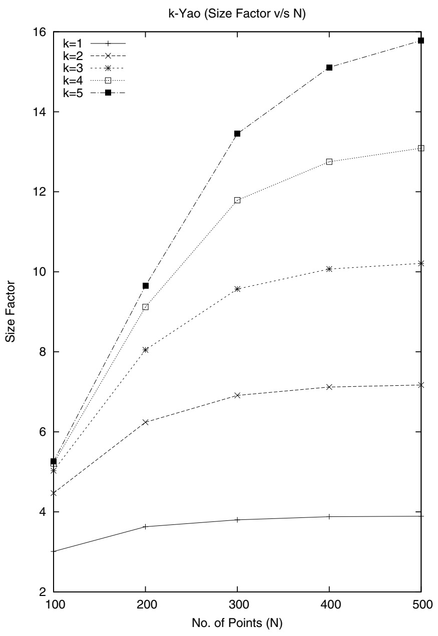
Fig. 5. Simulation results for *k*-Del



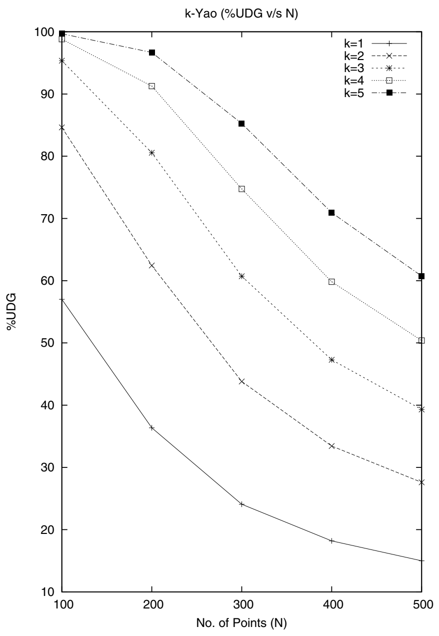
(a) Average Degree v/s Points



(b) Maximum Spanning Ratio v/s Points

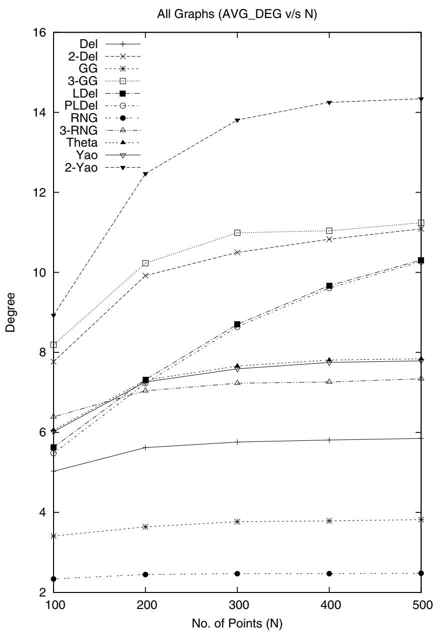


(c) Size Factor v/s Points

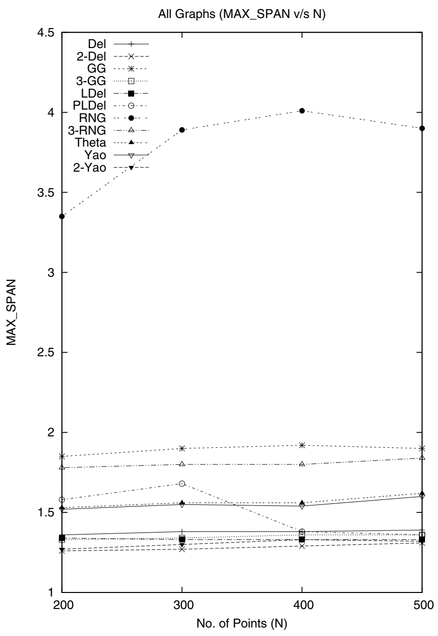


(d) Percentage of UDG v/s Points

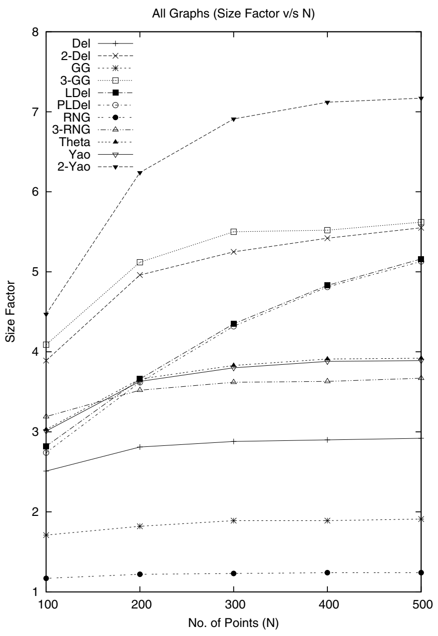
Fig. 6. Simulation results for *k*-Yao



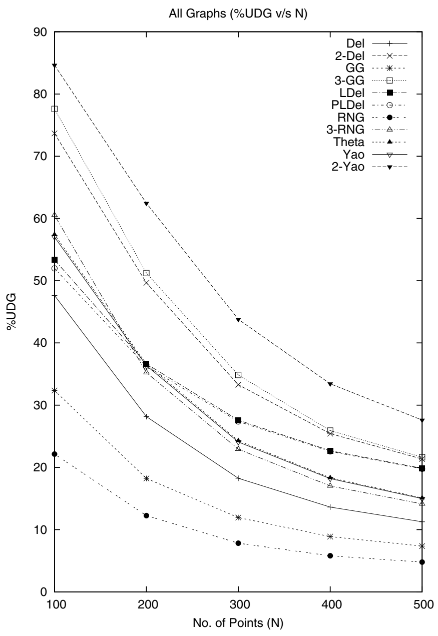
(a) Average Degree v/s Points



(b) Maximum Spanning Ratio v/s Points



(c) Size Factor v/s Points



(d) Percentage of UDG v/s Points

Fig. 7. Simulation results for various spanners

4 Conclusion

In this paper we have considered the basic geometric spanners used in ad-hoc networks and generalized to higher order graphs by relaxing the emptiness criteria for better topology control. It would be interesting to study various network parameters like delay, jitter, delivery ratio, and throughput, when these graphs used as network topology for routing.

References

1. Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press, New York (2007)
2. Cheng, X., Huang, X., Li, X.Y.: Applications of computational geometry in wireless networks (2003)
3. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recognition 12, 261–268 (1980)
4. Bose, P., Devroye, L., Evans, W., Kirkpatrick, D.: On the spanning ratio of gabriel graphs and beta-skeletons. SIAM J. Discret. Math. 20(2), 412–427 (2006)
5. Keil, J.M., Gutwin, C.A.: The Delaunay triangulation closely approximates the complete Euclidean graph. In: Dehne, F., Santoro, N., Sack, J.-R. (eds.) WADS 1989. LNCS, vol. 382, pp. 47–56. Springer, Heidelberg (1989)
6. Dobkin, D.P., Friedman, S.J., Supowit, K.J.: Delaunay graphs are almost as good as complete graphs. Discrete & Computational Geometry 5, 399–407 (1990)
7. Chew, P.: There is a planar graph almost as good as the complete graph. In: Proceedings of the Second Annual Symposium on Computational Geometry, SCG 1986, pp. 169–177. ACM, New York (1986)
8. Li, X.Y., Calinescu, G., Wan, P.J.: Distributed construction of planar spanner and routing for ad hoc wireless networks. In: INFOCOM (2002)
9. Chang, M.S., Tang, C.Y., Lee, R.C.T.: 20-relative neighborhood graphs are hamiltonian. In: Asano, T., Imai, H., Ibaraki, T., Nishizeki, T. (eds.) SIGAL 1990. LNCS, vol. 450, pp. 53–65. Springer, Heidelberg (1990)
10. Chang, M.S., Tang, C.Y., Lee, R.C.T.: Solving the euclidean bottleneck biconnected edge subgraph problem by 2-relative neighborhood graphs. Discrete Applied Mathematics 39, 1–12 (1992)
11. Chang, M., Tang, C., Lee, R.: Solving the euclidean bottleneck matching problem by k-relative neighborhood graphs. Algorithmica 8, 177–194 (1992), doi:10.1007/BF01758842
12. Rao, S.V., Mukhopadhyay, A.: Fast algorithms for computing beta-skeletons and their relatives. Pattern Recognition 34, 2163–2172 (2001)
13. Kiran, P.: kth order geometric spanners for wireless ad hoc networks. Master's thesis, Indian Institute of Technology Guwahati (2010)

Robust and Distributed Range-Free Localization Using Anchor Nodes with Varying Communication Range for Three Dimensional Wireless Sensor Networks

Manas Kumar Mishra and M.M. Gore

Department of Computer Science & Engineering
Motilal Nehru National Institute of Technology, Allahabad, India
{manasmishra,gore}@mnnit.ac.in

Abstract. Localization of the nodes in a sensor network is a premier activity which influences the performance of the network. The data collected by a sensor node may become useless if the location of that node is not known. Sensor networks are mostly deployed in areas where manual positioning of the sensor nodes is not feasible, and the topology and size of the network also changes frequently. Therefore, the localization schemes developed for these kinds of network need to be self-configurable and adaptive to the changes. This paper presents a localization scheme for three dimensional wireless sensor networks that not only helps sensor nodes to self-localize, but, it is also able to verify the estimated location and re-estimate it, if needed. The sensor nodes estimate their positions based on the position information of the GPS enabled anchor nodes. Simulation results show that the proposed method gains heavily in terms of the accuracy of the estimated positions.

1 Introduction

Of late, Wireless Sensor Networks (**WSN**) have been perceived as a befitting alternative in various critical applications like surveillance of terrains for landslide detection, invasion monitoring, study of underwater ecosystem for pollution monitoring, early warning systems for natural disasters like tsunamis, oil drilling, etc. Predominately these applications require three dimensional (**3D**) modeling of the WSN, as the event detection need to be associated with the height/depth of it's occurrence as well. The location information about occurrence of an event is estimated based on the location information of the participating nodes, detecting that particular event. Therefore, the localization algorithm is required to be designed to help sensor nodes to self localize. The condition gets more demanding when such sensor networks have to be deployed on a larger scale. Most of the terrestrial 3D localization schemes [3][6][10] use Global Positioning System (**GPS**) enabled anchor nodes for generation of beacon points to be used for localization. While the mobile anchor nodes 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 while the static node is at the surface of the communication sphere of the anchor node and before entering into it, and the second one is recorded at the surface again, while the node is about to leave the communication sphere. 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 (denoted as t) between recording of the two valid beacon points as described before, is given by the equation,

$$t = \frac{2 * \sqrt{r^2 - d^2}}{v} \tag{1}$$

Therefore, for a predefined speed of travel, 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. Further, despite the distribution of the sensor nodes being random in nature, the presence of patches of highly dense deployment can not be ruled out. In such a patchy distribution, if the diameter of a dense patch is much smaller than the communication range of the anchor node, then the time difference to receive two valid beacon points for the nodes in the patch will be too high, as most of the time the nodes will be with in the communication range of the anchor node. So, smaller communication range anchor nodes should be preferred. But, in contrast to this, the nodes located in a sparsely dense patch will have to wait for a longer period for the first valid beacon point if the communication range of anchor nodes are relatively smaller, and hence higher communication range should be used. Therefore, the use of anchor nodes with different communication ranges will be more efficient. But, the authors are unaware of any range-free scheme which uses anchor nodes with variable communication ranges for position estimation in 3D WSN. In addition to this, if the anchor nodes are battery (limited energy) operated, then the communication ranges of these nodes will get reduced over a period of time based on the number of beacon messages they broadcast, resulting in the variation of communication ranges for different anchor nodes. Further, the estimated position will inherit the fault in reading from the beacon points, if any, due to faultier GPS readings, wrongly estimated communication ranges, etc. Hence, localization techniques need to be developed to cater to the issues of error correction and varying communication range. This paper proposes a robust and distributed anchor based range-free localization algorithm for 3D WSNs that provides position estimations with accuracy using anchor nodes with variable communication range.

The paper is organized in five sections. In section 2, the related works are discussed. The proposed localization algorithm is elucidated in section 3, it also defines the network model and the assumptions made in the scheme. Section 4 presents and analyzes the simulation results. Finally, we conclude the paper in section 5.

2 Related Work

Range-free schemes [1][4][7][11] do not use any range measurements. Range-free methods are simple, but, give only a coarse estimation of the node location [5][9]. So, we require localization schemes that give accurate results as well as cost effective.

Marcelo Martins et.al proposed a localization scheme for 3D WSN in [6] using a centroid method. This method uses the location information of the anchor nodes to estimate the centroid of the tetrahedron composed of these points. The coordinates of the centroid is considered as the estimated position of the sensor node.

Chia-Ho Ou and Kuo-Feng Ssu in [3] proposed a range free scheme for 3D WSN which requires two chords to be built from the beacon points on the communication sphere. The chords need to have an angle of at least 10 degrees between them. This constraint results in a high probability of discarding the received anchor node positions. This scheme estimates the location of the sensor nodes with negligible errors.

A range-free localization scheme for 3D WSN is proposed in [10] by V.Yadav et.al. This approach uses basic principle of 3D geometry for location estimation. With four anchor positions, four equations of sphere can be formed using which we can find the three coordinates of the node location in a three dimensional space. This scheme requires only one non-coplanar beacon point among the four positions received. This lowers the probability of discarding any anchor position as compared to the approach mentioned in [3].

The schemes proposed in literature do not verify the estimated position, which may have in it the inherited errors due to the calculations made, or faulty GPS readings, if any. Further, the usefulness of varying communication range for location estimation of nodes in a patchy deployment has not been explored. In addition, the communication ranges of anchor nodes can also reduce due to the depletion of their battery power, if it is not assumed to be unlimited. Our proposed localization algorithm solves these problems.

3 Proposed Localization Algorithm

3.1 Network Model and Assumptions

Types of Nodes: The WSN model used for this scheme consists of two types of nodes: sensor nodes and anchor nodes. Sensor nodes are large in number and will be used for sensing the data. Anchor nodes are few in numbers and are equipped with the GPS device, hence, they are able to find their own exact positions. These nodes also have more battery power than the static nodes as these have to continuously transmit the beacon messages. All the nodes of the network are randomly deployed.

Node Property: The communication ranges of all the nodes are assumed to be spherical. The sensor nodes have a fixed communication range. 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 RF 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. In this work, the mobility model for these kind of nodes are considered to be that of Random Walk, and Random Way Point [2].

3.2 Beacon Point Selection

As the anchor nodes move, they continuously broadcast their positions and their current communication range. The static unlocalized nodes listen to all these messages, but stores 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 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.

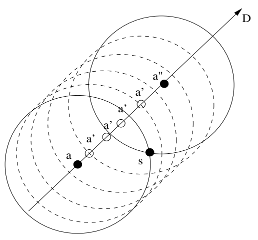


Fig. 1. Valid beacon points a and a'' received by s

3.3 Position Calculation

When any sensor node registers a valid beacon point, then the sensor node can be assume 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

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \tag{2}$$

Let (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) and (x_4, y_4, z_4) be the position coordinates received from four anchors with communication range as r_1, r_2, r_3 and r_4 respectively. Using these values, four equation of sphere with the sensor node at the center of each sphere can be formed as

$$(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = r_1^2 \tag{3}$$

$$(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = r_2^2 \tag{4}$$

$$(x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = r_3^2 \tag{5}$$

$$(x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2 = r_4^2 \tag{6}$$

Above equations can be solved for the values of x, y and z as follows,

$$x = \begin{bmatrix} ((x_2^2 + y_2^2 + z_2^2 - r_2^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \; y_2 - y_1 \; z_2 - z_1 \\ ((x_3^2 + y_3^2 + z_3^2 - r_3^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \; y_3 - y_1 \; z_3 - z_1 \\ ((x_4^2 + y_4^2 + z_4^2 - r_4^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \; y_4 - y_1 \; z_4 - z_1 \end{bmatrix} / \Delta \tag{7}$$

$$y = \begin{bmatrix} x_2 - x_1 \; ((x_2^2 + y_2^2 + z_2^2 - r_2^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \; z_2 - z_1 \\ x_3 - x_1 \; ((x_3^2 + y_3^2 + z_3^2 - r_3^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \; z_3 - z_1 \\ x_4 - x_1 \; ((x_4^2 + y_4^2 + z_4^2 - r_4^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \; z_4 - z_1 \end{bmatrix} / \Delta \tag{8}$$

$$z = \begin{bmatrix} x_2 - x_1 \; y_2 - y_1 \; ((x_2^2 + y_2^2 + z_2^2 - r_2^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \\ x_3 - x_1 \; y_3 - y_1 \; ((x_3^2 + y_3^2 + z_3^2 - r_3^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \\ x_4 - x_1 \; y_4 - y_1 \; ((x_4^2 + y_4^2 + z_4^2 - r_4^2) - (x_1^2 + y_1^2 + z_1^2 - r_1^2))/2 \end{bmatrix} / \Delta \tag{9}$$

where

$$\Delta = \begin{bmatrix} x_2 - x_1 \; y_2 - y_1 \; z_2 - z_1 \\ x_3 - x_1 \; y_3 - y_1 \; z_3 - z_1 \\ x_4 - x_1 \; y_4 - y_1 \; z_4 - z_1 \end{bmatrix} \tag{10}$$

3.4 Error Detection and Correction

The major contribution of the paper 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 from 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 carried out. The error correction can be achieved by using any of the three different approaches.

1. The simplest approach is to discard all the four beacon points and rerun the algorithm for the estimation of the location coordinates of the static node, again. This method takes more time as it need to record four valid beacon points again.
2. Another approach is to discard the beacon point corresponding to the anchor node having the largest communication range amongst the four beacon points recorded. The reason for the selection of the largest communication range is that, in presence of interference, the actual distance between the anchor node and the static sensor node will be most effected for the largest communication range. In case of a tie between beacon points on this criteria, any one of them may be dropped.
3. The modified version of the second approach is the weighted approach for determining the beacon point to discard. If the error is found to be positive that is if the distance is larger than the communication range, then the actual position of the static node is nearer to the anchor node instead of the estimated position. Hence, the farthest beacon point form the anchor node on the same side as that of the static node is the heavy weight in the contribution to the error detected. Therefore, it should be dropped from the record, and the position is re-estimated taking the other three beacon points and the current position of the anchor node as the fourth beacon point. But, if the error is found to be negative that is the distance is smaller than the communication range, then the actual position of the static node is farther to the anchor node instead of the estimated position. Thus, the farthest beacon point from the anchor node on the opposite side of the static node is the heavy weight in the contribution to the error detected. Therefore, it should be dropped for the record, and the position is re-estimated taking the other three beacon points and the current position of the anchor node as the fourth beacon point. The direction of the beacon points are estimated using angle between the line joining the estimated coordinates of the static node and the position of the anchor node, and the line joining the anchor node and the beacon points, respectively. Positions having angles less than 90^0 are considered on the same side, and positions having the angles greater than 90^0 are considered to be on the opposite side. In case of a tie between beacon points on this criteria, any one of them may be dropped. But, if no beacon is identified as the candidate to be dropped based on this criteria, then the anchor node is considered as ineffective for error correction. The static node is considered to remain in estimated state and waits for an effective fifth beacon position for error correction.

In both the second and the third approaches, the gain is due to the dropping of only one beacon point as compared to the first approach where all the four beacon points are dropped. But, the first method has an added advantage that, it can relocate the accidentally displaced nodes, whereas the other two methods can only make error corrections to the estimated location information.

3.5 Algorithm

The proposed method uses the GPS enabled anchor nodes to know their position and broadcast these along with their communication range. The position of an unlocalized sensor node is calculated based on the position information in the messages received from the anchor nodes. The anchor nodes use Random Walk, and Random Way Point as mobility models. Algorithm 1 presents the detailed algorithm for localization of the static sensor nodes.

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
```

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 position from the SET and again waits for the beacon message to get a new node position. If the determinant is not equal to zero, then location can be found. After the location is found, a new position is needed for verification of the estimated location. 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 section IV D. The algorithm terminates once all the nodes are localized.

4 Simulation Results

The proposed approach is simulated on Sinalgo-0.75.3-Regular Release [8]. It provides a simulation framework for testing and validating network algorithms in both two and three dimensions. Sinalgo offers a broad set of network conditions.Table 1 presents the simulation settings applied for the execution of the proposed algorithm.

Table 1. Simulation Settings

Total Number of Sensor Nodes	525
Static Sensor Nodes	500
Mobile Sensor Nodes	25
Simulation Area: Dimensions:	3 (500mX500mX500m)
Simulation:	
Synchronous Mode	True
Interference Model	No Interference
Connectivity Model	UDG (rmax=175,150,125)
Reliability Model	Reliable Delivery
Distribution Model	Random(for Static Nodes and Mobile Nodes)
Mobility Model	No Mobility(for Static Nodes), Random Walk/Random Way Point(for 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)

We compared the results obtained using different mobility models. The results obtained by implementing the method using fixed and variable communication ranges over uniform distribution has been compared. We have also shown the effect of variable communication range method in case of non-uniform distribution leading to patchy dense regions. The results obtained using three different error correction methods have also been analyzed. Further, two parameters defined below have been used to present and analyze our results with the results obtained using the method proposed in [3].

– *Average localization time*: The average time required for all sensor nodes to compute their locations i.e.

$$AverageLocalizationTime = \frac{\Sigma Localization\_time}{no.of sensors} \tag{11}$$

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.

$$AverageLocationError = \frac{\Sigma\sqrt{(x_e - x_L)^2 + (y_e - y_L)^2 + (z_e - z_L)^2}}{no.of sensors} \tag{12}$$

It checks the accuracy of the computed locations.

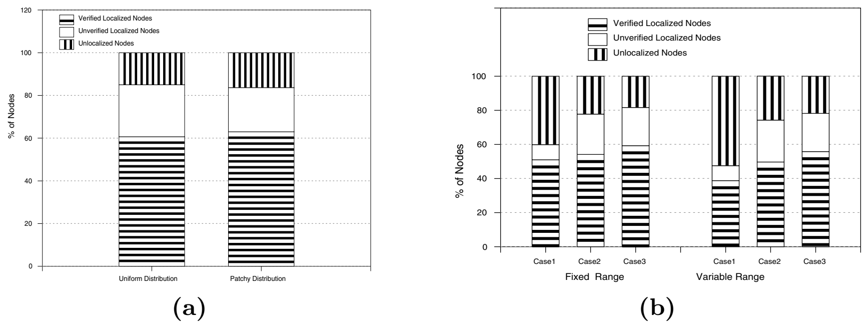


Fig. 2. Performance for a sample run of 500 rounds using Random Way Point mobility model. (a) Variable communication range using different distribution models (b) Fixed and variable communication range with uniform distribution.

4.1 Fixed vs. Variable Communication Range

We advocated in the previous sections that use of variable communication range will help in applications having patchy distributions and anchor nodes having limited power supply. The implementation of this method has been compared for uniform distribution and patchy distribution in Figure 2a. The results show that the performance is better in patchy distribution as compared to uniform distribution condition. We also present the results for implementation with fixed communication range and variable communication range under uniform distribution using Random Way Point mobility model, in Figure 2b. The cases in the results refer to the three error correction methods, respectively. The two results in Figure 2 show that, there is not much of performance difference between fixed and variable communication range methods in uniform distribution, whereas, the variable communication range method gains in performance when implemented in patchy distribution.

4.2 Mobility Model

The results for different mobility models using variable communication range have been compared for all the three cases of error correction. Figure 3 shows the results. The results show that the Random Way Point mobility model using the third error correction method has the best performance.

4.3 Error Correction

We also compare the error correction methods in Figure 4a. The results are obtained implementing variable communication range method with different mobility models using all three error correction methods. The results show that, the accuracy in localization is best in case 1, whereas, the results in case 3 is better than case 2. As discussed in earlier sections, the first method of error correction gains in accuracy at the cost of time of localization. Figure 4b presents

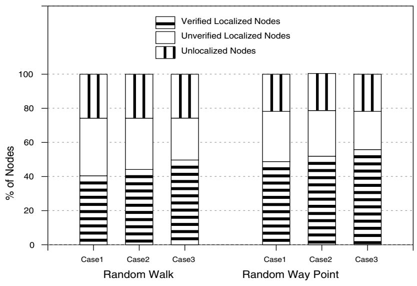


Fig. 3. Performance using variable communication ranges under different mobility models for a sample run of 500 rounds with uniform distribution

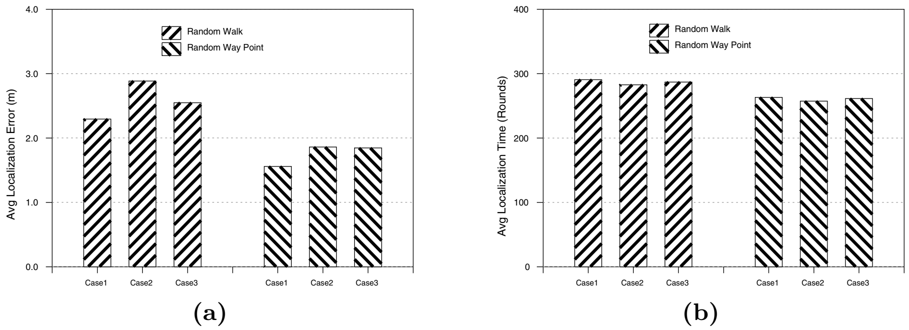


Fig. 4. Performance using variable communication range with different error correction methods for a sample run of 500 rounds under uniform distribution model. (a) Average Localization Error (b) Average Localization Time.

the average localization time for all the three error correction methods. The case 2 takes relatively less time as discarding all the beacon points makes the method slow in case 1, whereas the calculations to be made to determine the candidate beacon point to discard in case 3 results in the method taking more time.

4.4 Average Localization Time

The average localization time as defined earlier is used to compare the results obtained from the approach in [3] with that of our approach. We conducted simulations with 250 static nodes to find out the total localization time in terms of rounds, using both the approaches. Figure 5 presents the results between proposed approach and approach in [3] with 25 anchor nodes being deployed, and using both Random Walk and Random Way Point as the mobility models. 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 6a presents the average localization time for both the approaches using the two mobility models.

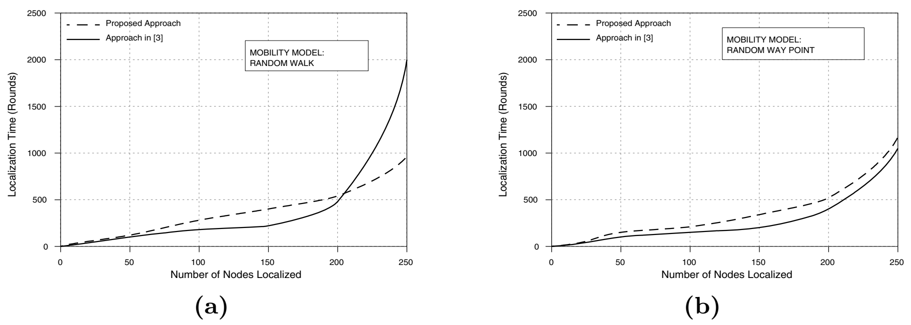


Fig. 5. Performance comparison on Localization time

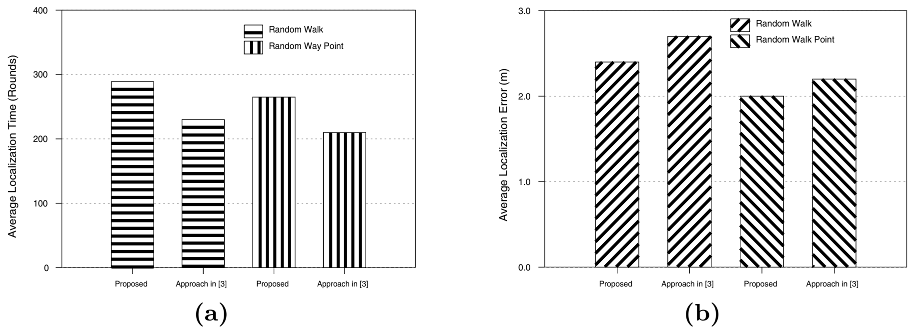


Fig. 6. Performance comparison for a sample run of 500 rounds. (a) Average localization time (b) Average localization error.

4.5 Robustness (Average Localization Error)

The average localization error as defined earlier is also used to compare the results obtained from both the approaches. Figure 6b presents the average localization error for the approach in [3] with the proposed method using the third error correction approach. The results show that, though the proposed approach takes more time to get the nodes localized, its accuracy of localization is a major advantage.

5 Conclusion and Future Work

The presented scheme not only localizes sensor nodes but, also verifies the accuracy of the estimated locations. Displaced nodes are able to detect the change in their location and can relocate themselves using the first error correction method. This scheme requires more beacon messages to be transmitted, but, the simple implementation of the scheme makes it a much better option. The accuracy of the estimated locations is better than other range-free schemes. The relocation of any node can easily be done if detected within the total localization time. But, once localization of all the nodes are over the anchor nodes need to move again to detect the change in position of nodes and to subsequently relocate them. The adaptability of the method for the variable communication range makes it more suitable for deployments leading to patchy regions. In addition, this method can accommodate the change in the communication range of the anchor nodes due to power depletion, if any. Detection of changed position and relocation of any node using the other localized static nodes can be considered as a future work. A tolerance limit for the mobility of the static nodes in terms of its transmission range can also be considered as a future scope of work.

Acknowledgements. The authors would like to thank Ms. Neha Ojha for her support in the simulation of the approach.

References

1. Blum, B.M., Stankovic, J.A., Abdelzaher, T., He, T., Huang, C.: Range-Free Localization and Its Impact on Large Scale Sensor Networks. ACM Transactions on Embedded Computing Systems (TECS), 877–906 (2005)
2. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.-C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocols. In: Proc. of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1998), ACM, New York (1998)
3. 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)
4. Niculescul, D., Nath, B.: DV Based Positioning in Ad Hoc Networks. SpringerLink Journal of Telecommunication Systems, 267–280 (2004)
5. Mao, G., Fidan, B., Anderson, B.D.O.: Wireless sensor network localization techniques. Computer Network: The International Journal of Computer and Telecommunications Networking 51, 2529–2553 (2007)
6. 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)
7. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less Low Cost Outdoor Localization For Very Small Devices. IEEE Journal of Personal Communications, 28–34 (2000)
8. Sinalgo: Simulator for network algorithms (2009), http://dcg.ethz.ch/projects/sinalgo
9. He, T., Stoleru, R., Stankovic, J.A.: Range free localization. Technical report, University of Virginia (2006)
10. Yadav, V., Mishra, M.K., Singh, A.K., Gore, M.M.: Localization Scheme for Three Dimensional Wireless Sensor Networks Using GPS Enabled Mobile Sensor Nodes. International Journal of Next-Generation Networks 1(1) (2009)
11. Vivekanandan, V., Wong, V.W.S.: Concentric Anchor-Beacons (CAB) Localization for Wireless Sensor Networks. IEEE Transactions on Vehicular Technology, 2733–2744 (2007)

Decision Support Web Service

N. Parimala and Anu Saini

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India
dr.parimala.n@gmail.com,
anuanu16@gmail.com

Abstract. Currently, UDDI (Universal Description Discovery and Integration) is a standard for publishing and discovery of Web services. The UDDI search mechanism is based on functional properties and the consumer chooses a web service based on these properties. When functional properties are identical, additional distinguishing characteristics are needed to choose a service. Towards this, we propose to associate non functional properties, referred to as criteria, with a web service. A web service that has a list of criteria associated with it is termed as Decision Support Web Service (DSWS). In order to associate criteria, the UDDI is extended to X-UDDI which includes a new orange page containing the list of criteria and their description. We also add one more bag, criteria bag, which is used to define the attributes of the criteria. In order to publish and invoke a service on both, the functional properties and the criteria, we define new APIs.

Keywords: Web service, Decision support web service (DSWS), Non functional properties, UDDI, Extended UDDI.

1 Introduction

Service Oriented Architecture (SOA) is a business architecture which provides business functionality and application logic to users or consumers as a shared and reusable service [1, 2]. Main building blocks of SOA are these reusable services.

Web service is defined as a piece of business logic which is available on the internet and that can be accessible with the help of internet protocols [3, 4].

Web service has three characteristic: functional, behavioural and non-functional [5]. Functional description tells us about what exactly the service can do. Behavioural description details how the web service works and how it can be integrated using Orchestration and Choreography in WSMO (Web Service Modeling Ontology). Non-functional descriptions are the constraints on the functional properties which are given by the user to discover the service.

Normally, functional characteristics and also some non-functional parameters like provider name, service name and category are used to search the registry. Using just these characteristics for search may be inadequate in some cases. For instance, consider the case where the user might like to decide dynamically in which hotel he wishes to stay. The choice may not necessarily depend on the basic information like name of the hotel, but may depend on some other facilities like swimming pool, room service etc. In this case it is necessary that facilities on which a choice can be made be defined and made available with a web service.

The idea of associating additional information with a service to help locate the right service has been addressed by many researchers. In [6] this additional information has been specified by extending the UDDI (Universal Description Discovery and Integration) to UDDIe. [9] has provided non functional properties as part of the tmodels. Semantic web services do not exactly add additional properties but associate meaning with a web service thereby helping the consumer identify the service.

In this paper, non-functional properties are associated with the service and these properties form the criteria for the user to choose a service. Such a web service is termed 'Decision Support Web Service' (DSWS). It is termed as 'Decision Support' because it helps the client to decide whether a service is to be invoked or not based on functional and non-functional properties.

To store more information we extend the traditional UDDI to the X-UDDI by adding one more page (orange page). The page is used to save the criteria which are associated with the service. The page is flexible and can be easily modified and updated by the provider. Key references of the service are associated with criteria. To store these, we introduce a new bag called the Criteria Bag. We also add the necessary APIs to publish and invoke the criteria associated service. We introduce an attribute, criteriaMatch which allows for full or partial match of the user specified criteria with the criteria stored in the X-UDDI.

The layout of the paper is as follows. A survey of related work is dealt with in section 2. Section 3 defines a decision support web service and the architecture of our proposals. In section 4 the extension of UDDI is explained. Section 5 describes the extended APIs of X-UDDI. An example is given in section 6. Section 7 is the concluding section.

2 Related Work

As mentioned above, the ability of the user to differentiate between the services which provide the same functionality and select the right one depends upon their non-functional properties. The non-functional properties that can be specified are quality of service (QoS), performance, scalability, reliability, availability, stability, cost, completeness etc [7]. A number of attempts have been made to describe and discover service based on the non-functional properties. We describe these methods in detail below.

1. Extension of UDDI to UDDIe is used to include QoS attributes for discovering and describing the service [6, 8]. The UDDI registry is extended to UDDIe wherein the attributes of a service are stored. In [6] the client requests for a service with associated additional attributes. The condition on the attributes can be combined using Boolean operators AND/OR. The QoS information is stored in the blue page and is created when the service is published. However, since the information is in the blue page, it cannot be

updated to incorporate any changes in QoS. Further, the service is stored in the UDDIe for a finite period of time and hence does not support the reusability of service.

2. Quality of service (QoS) which includes performance, reliability, availability and throughput are defined in tmodels as a non-functional property [9]. tmodels represents concepts or constructs that are used to describe compliance with a specification. Four different methods for storing the QoS in UDDI are used by the tmodels [9, 10]. In the first method the QoS tmodel, called QoSinformation, which references an external quality of service, is defined in the UDDI. Each UDDI bindingTemplate contains a QoSInformation tModel, and adds the QoSInformation tModel to the tModelInstanceDetails collection. The second method creates many additional tmodels for different QoS information. These categories are added to the binding templates. The third method is similar to the first method. The only difference is that it contains the binding template. As a result, the category bag of QoSinformation tmodel has many key references to represent different QoS information. The fourth method stores the QoS values in the categoryBag of businessService in UDDI. All these methods require multiple steps to locate the desired service. As a result the time required to locate a service would be long. It can lead to an inefficient system. These methods also contain the unmanageably large category bags with many categories on each binding template. These methods are, therefore, difficult to use to manage the non-functional properties.
3. Semantic Web Service (SWS) [11, 12] provides meaningful information to help the user in searching, discovery, selection and composition of web service. The semantics of the web service is treated as the non functional property of the web service. It is described in the ontology. The non-functional properties considered in semantic web service ontology include service author, service contact information, service contributor, service contributor, description, service URL, service identifier, version, release date, language, trust, subject reliability and service cost. The service is discovered by matching the description with the user requirement. A different level of matching is used in [13] like exact, subsume, plug-in, intersection and disjoint. In SWS the attempt is to discover the web service automatically with the help of ontologies. Even though semantic web service has major benefits, there are some drawbacks. Ontologies are not flexible, difficult and complex to define [14, 15]. Making a consistent and appropriate ontology and its further maintenance is another big problem in the usage of semantic web service [15]. Further, it is difficult for the user to understand the semantics as they are complex and difficult to use. Another problem is that semantic web service lacks repositories or SWS marketplace where interaction could take place [15].
4. Web service Peer-to-Peer discovery service (WSPDS) is also used for discovery of service based on non-functional properties which includes interoperability, scalability, efficiency, fault tolerance and semantics [16, 17, 18]. These Peer-to-Peer systems, referred to as unstructured systems, are decentralized, scalable, and self organizing. In Peer-to-peer discovery of

service, the role of provider is eliminated [17]. All system entities work together in a distributed manner to provide a service. Here, each entity is work as a server and client of the peer-to-peer service. In this system there is no way to determine which peer in the system is more likely to have certain data. Thus, it leads to inefficient search.

5. Agent framework for discovery of Web services with QoS is proposed for determining the best service according to the user requirement [19, 20,21]. In this, both, service consumers and provider participate via the agent framework. QoS data that is collected from agents is stored, aggregated and then shared between them. The agent-based framework is implemented using the Web Services Agent Framework (WSAF). In [20], QoS ontology is used to capture and define the most generic quality concepts. On the other hand, the framework proposed in [19] contains many non-functional properties like reliability, availability, and request-to-response time.
6. QoS Broker Based approach is used for dynamic selection of the web service [22, 23]. This approach is used to specify the non-functional requirements as well as the functional requirements. QoS broker is placed between the web service registry and service requester who helps the requester to specify and select the appropriate web service according to his/her requirement. In [23], QoS is used to select and rank the web service. In this approach a tree model is made for the requester's QoS requirements by considering the QoS request.

In our approach we associate a list of attributes with a service. The client searches for a service having a desired set of attributes. The service which matches his/her request is returned via SOAP protocol. Even though we extend UDDI to X-UDDI, we are different from [6] in that in our architecture the result of a search is directly handled by the user and there is no broker to decide which is the best service that meets the client's needs. This is in keeping with our philosophy that we provide a decision support service and the ultimate decision is to be taken by the client. We are different from semantic web service as we do not associate any meaning to a service but enhance the service with non functional properties. As opposed to the usage of tmodels, our approach is a single step approach.

3 Architecture

Decision support web service (DSWS) is defined as follows:

'A decision support web service is a service with which multiple criteria are associated'

For deploying DSWS, we need to explore the technical aspects of publishing and invoking the service. Currently for web services, Universal Description Discovery and Integration (UDDI) registry is used to publish WSDL (Web Service Description Language) document containing the web service description, so that users can access the service.

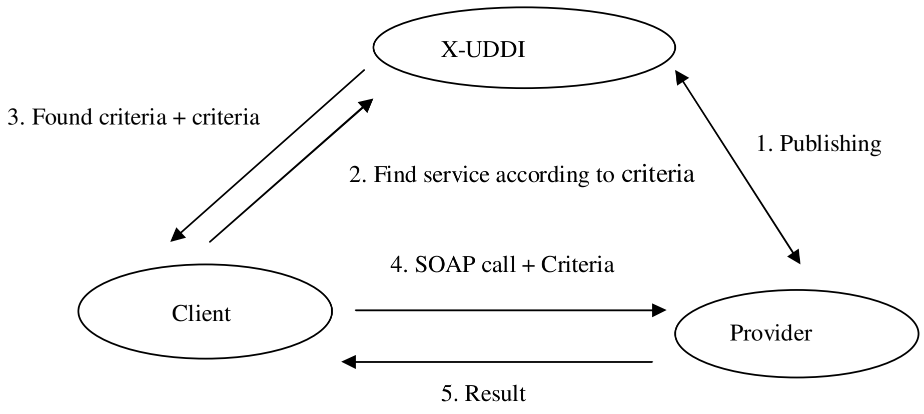

X-UDDI
3. Found criteria + criteria
1. Publishing
2. Find service according to criteria
4. SOAP call + Criteria
Client
Provider
5. Result

Fig. 1. Modified Architecture of SOA with X-UDDI

The architecture of our system is shown in Figure 1. The steps are

1. The provider saves the DSWS in the X-UDDI with the description of its criteria.
2. Client requests for a web service with a criteria list.
3. The matching services are returned by the X-UDDI.
4. Client chooses one among the list that was received and sends a SOAP call to the provider.
5. Provider sends the result to the client.

4 X-UDDI

In this section we explain the extensions to UDDI. Before doing so, we briefly describe the UDDI structure which is relevant to the extension.

UDDI is an XML-based registry for businesses worldwide to list themselves on the Internet [24, 25]. The four data structures of UDDI are business entity, business service, binding template, and tmodel. UDDI consists of pages which are associated with each data structure. Existing UDDI consists of white, yellow and green pages. White page is associated with business entity. These pages hold the basic contact information of the company. Yellow page is associated with business service. These pages organize web services into categories like usage billing service, authorization service and so on. Binding template is associated with green page. Green pages provide detailed technical information about the individual services.

As can be seen from the above, there is a need to introduce a new page to store the information of each criterion of this service. Thus, we extend UDDI to X-UDDI where X-UDDI contains the additional page, referred to as the ‘Orange Page’. The orange page contains the criteria and the associated detailed information. As defined in [6] every service has its details described in business service. Therefore, we add the criteria information to the Business service structure as shown in Figure 2. Since the details of the criteria are in the orange page, we associate the business service with the newly defined orange page. This is in addition to the already existing association with yellow page.

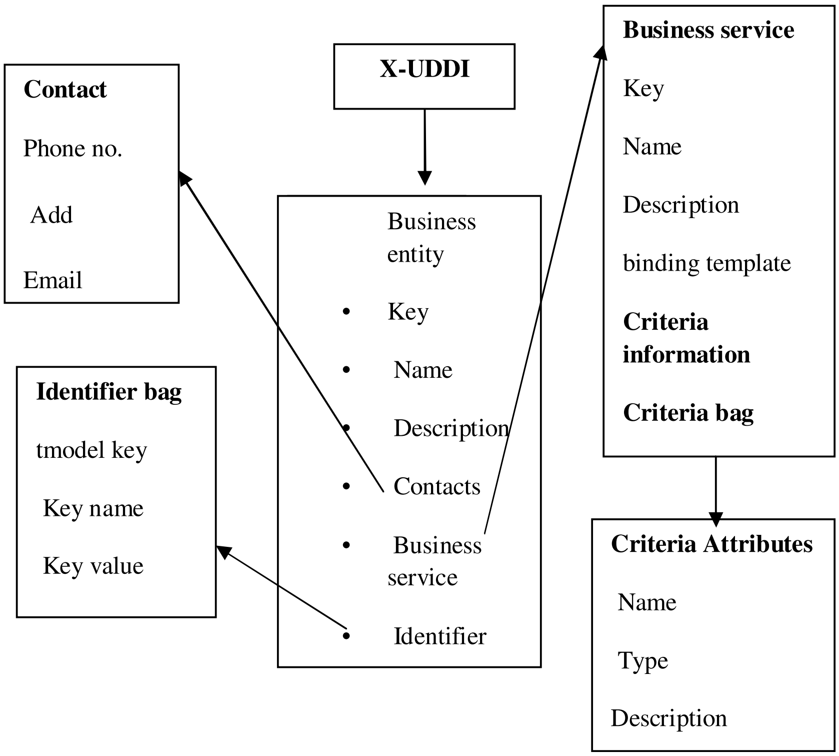


Fig. 2. X-UDDI Structure

We introduce a Criteria Bag in the X-UDDI which contains the key reference of the DSWS and the criteria attributes. The criteria attributes associated with each criterion are name, type and a description as shown in Figure 3.

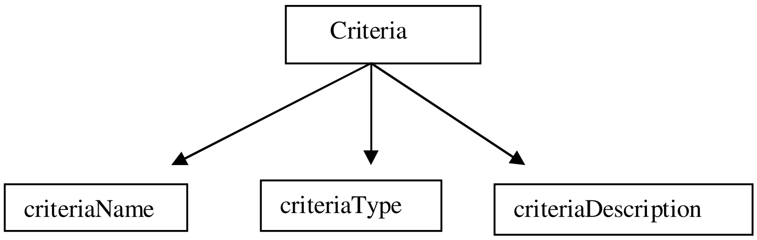


Fig. 3. Criteria Attributes

An example of the criteriaBag is given below.

```
<criteriaBag>
<criteria>
      <criteriaName>SwimmingPool</criteriaName>
      <criteriaType>String</criteriaType>
      < criteriaDescription> "Swimming pool facility is
      present" </ criteriaDescription>
</ criteria >
</ criteriaBag>
```

5 APIs of X-UDDI

So far we have described the extensions to UDDI. Now, we need APIs (Application Programming Interface) to store the orange page and the criteria bag in the X-UDDI. We also have to define APIs to access the DSWS. These are explained below.

The programming interface for UDDI consists of two parts: a Publishing API and an Inquiry (search) API.

The Publishing API is for the use of service providers. Publishing interface consists of save_business, save_service, save_binding, save_tModel, delete business, delete_service, delete_binding, delete_tModel.

The Inquiry API is used by the clients to access the web service. Inquiry interface consists of find_business, find_service, find_binding, find_tModel, get_businessDetail, get_serviceDetail, get_bindingDetail, get_tModelDetail.

In X-UDDI we have extended both the publishing as well as the inquiry APIs. Publishing API is extended by defining save_dservice interface. Find_dservice is the part of inquiry interface. We consider below first the publishing API followed by the inquiry API.

5.1 Publishing API of DSWS

After developing the DSWS we have to publish it in the X-UDDI. We extend the existing UDDI API by adding the following publishing API:

save_dService: save_dservice is used to publish DSWS. In addition to the entries defined in Save_service API, save_dService has an additional entry which is the criteria bag with the list of criteria. The criteria information along with other information pertaining to the service is stored in the X-UDDI.

5.2 Inquiry API of DSWS

The inquiry API, find_dservice, allows the user to locate and obtain the service which matches the criteria list.

find_dService: It has the criteria bag with the desired criteria list and an optional matching option. The other entries of find_dService are as defined in find_service API. The matching option, criteriaMatch, has two values. These are full_criteria_match and partial_criteria_match. In the case of full_criteria_match, the service returned by find_dService will have exactly the same criteria as passed by the client. In the case of partial_criteria_match, the service returned by find_dService will have a minimum of one criterion among those specified by the client.

6 Example

Consider an example of booking a room in the hotel which offers multiple criteria. Suppose Taj_Hotel_Service is a DSWS which is associated with three criteria which are

1. Room Service,
2. Swimming Pool and
3. Internet access.

6.1 Publishing the Service in the X-UDDI

In this example, Taj-Hotel_Service is providing three facilities which are to be associated as criteria with the service. The code for publishing Taj-Hotel_Service with criteria list is given below. The code below depicts only the manner in which we associate criteria with a service and the other details like binding template are not shown as these details are specified as per the WSDL format.

```
<save_dService generic="2.0" xmlns="urn:uddi-org:api_v2">
<businessService businessKey="*****" serviceKey="">
        <name>Taj_Hotel_Service</name>
        <criteria Bag>
          < criteria >
            < criteria Name>Room_service</ criteria Name>
            < criteriaType>string</ criteriaType>
            < criteriaDescription> " Facility of room
             service " </ criteriaDescription>
          </criteria >
        < criteria >
          <criteriaName>Swimming_Pool</ criteria Name>
          < criteriaType>string</ criteria Type>
          < criteriaDescription> "Swimming pool facility
           is present "</ criteriaDescription>
     </ criteria >
     < criteria >
          <criteria Name>Internet_Access</ criteria Name>
          < criteria Type>string</ criteria Type>
         < criteriaDescription> "Internet facility is
          available"</ criteriaDescription>
     </ criteria >
    </ criteriaBag>
</BusinessService>
</save_dService>
```

6.2 Finding a DSWS

Let us say that the client wants to find a hotel service which has room service and an internet connection. Both the criteria must be available. Therefore, the client must chooses Full_criteria_match. The snippet of the call is given below. Only the details pertaining to find_dservice is given in the code below.

```
<find_dServicebusinessKey="*****"generic="2.0"
xmlns="urn:uddi.org:api_v2">
    <name>Hotel</name>
<categoryBag>
   <keyedReferencetModelKey="******"keyName="******"
  keyValue="******" />
</categoryBag>
    < criterialBag>
       <criteriaMatch>Full_criteria_match</criteriaMatch>
        < criteria>
           < criteriaName>Room service</criteriaName>
           <criteriaDescription>"Room service facility is
           available or not" </criteriaDescription>
        </ criteria >
        < criteria >
               <criteria Name> Internet </ criteria Name>
              <criteriaDescription> "Internet facility is
               available or not" </ criteriaDescription>
        </ criteria >
     </ criteriaBag>
</find_dService>
```

7 Conclusion

In this paper we have extended the web service to decision support web service where a DSWS has a list of criteria associated with it. This helps the client to search for a service which suits his needs.

We have extended UDDI by adding a new orange page, which contains criteria list along with their description. We have also added one more bag, criteria bag, which is used to define the attributes of the criteria. When the service is searched for, the user has the option of specifying whether there can be a partial match or whether there must be a full match between the attributes specified by the user and those stored with the service. Towards this, we have introduced the attribute criteriaMatch with two values, namely fullCriteriaMatch and partialCriteriaMatch. We have introduced APIs for publishing and for inquiry. save_dService is used for publishing the service whereas find_dservice allows to search for a service.

In this work, a single service which matches the criteria of the client is returned. We propose to extend this to handle more than one service in a business workflow.

Here, we have not considered the effect of scaling down X-UDDI to standard UDDI as and when required. This issue will be addressed is our subsequent work.

Note that in this paper we expect the user to specify the name of the criteria to be identical to that given with the service. It may be possible that the user specifies, not the identical name but a name which has the same semantics. This issue of semantically equivalent criteria is outside the scope of our work.

References

1. MacVittie, L.: Aligning Application Infrastructure with Business through Service-Oriented Application Delivery. F5 Networks, Inc. (March 2007)
2. Proteans, white paper.: Service Oriented Architecture Provide Flexibility to Business Operations. Proteans Software Solutions Pvt Ltd Copyright (2009)
3. Chappell, D., Jewell, T.: Java Web Services, 1st edn. O'Reilly, Sebastopol (March 2002)
4. Srivastava, B., Koehler, J.: Web Service Composition - Current Solutions and Open Problems. In: ICAPS (2003)
5. Toma, I., Foxvog, D.: Non-functional properties of web services. WSMO working draft (October 25, 2006)
6. ShaikhAli, A., Rana, O.F., Al-Ali, R., Walker, D.W.: UDDIe: An Extended Registry for Web Services. In: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT 2003 Workshops). IEEE Computer Society, Washington (2003)
7. Chen, Y.-p., Li, Z.-z., Jin, Q.-x., Wang, C.: Study on qoS driven web services composition. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 702–707. Springer, Heidelberg (2006)
8. Garcia, D.Z.G., de Toledo, M.B.F.: A Uddi Extension For Business Process Management Systems
9. Blum, A., Carter, F.: Representing Web Services Management Information (2004), `http://www.oasis-open.org/committees/../UDDI%20WSM-Info-1v7.doc`
10. Blum, A.: UDDI as an Extended Web Services Registry. Soa World Magazine
11. The OWL Services Coalition. OWL-S specification version 1.2 (November 2006), `http://www.daml.org/services/`
12. Shakya, A., Takeda, H.: Information Sharing on the Social Semantic Web. Springer, Heidelberg (2004)
13. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proceedings of the Twelfth International Conference on World Wide Web (WWW 2003), pp. 331–339. ACM Press, New York (2003)
14. Al Hunaity, M.A.: Towards an Efficient Quality Based Service Discovery Framework. In: IEEE Congress on Services (2008)
15. Haniewicz, K., Kaczmarek, M., Zyskowski, D.: Semantic Web Services Applications- A Reality Check. Gabler Verlag, Wirtschaftsinformatik (2008)
16. Emekci, F., Sahin, O.D., Agrawal, D., El Abbadi, A.: A Peer-to-Peer Framework for Web Service Discovery with Ranking. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2004 (2004)
17. Banaei-Kashani, F., Chen, C.-C., Shahabi, C.: WSPDS: Web Services Peer-to-peer Discovery Service. University of Southern California, Los Angeles (2004)
18. Vu, L.-H., Hauswirth, M., Aberer, K.: Towards P2P-based semantic web service discovery with qoS support. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 18–31. Springer, Heidelberg (2006)

19. Maximilien, E.M., Singh, M.P.: A Framework and Ontology for Dynamic Web Services Selection. In: IEEE Educational Activities Department Piscataway, NJ, USA (2006)
20. Maximilien, E.M., Singh, M.P.: A Framework and Ontology for Dynamic Web Services Selection. IEEE Internet Computing 8(5), 84–93 (2004)
21. Rajendran, T., Balasubramanie, P.: An Efficient Framework for Agent-Based Quality Driven Web Services Discovery. In: lAMA IEEE (2009)
22. Rajendran, T., Balasubramanie, P., Cherian, R.: An Efficient WS-QoS Broker Based Architecture for Web Services Selection. International Journal of Computer Applications 1(9) (2010)
23. D'Mello, D.A., Ananthanarayana, V.S., Thilagam, S.: A QoS Broker Based Architecture for Dynamic Web Service Selection. In: Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS), May 13-15, pp. 101–106 (2008)
24. Brittenham, P.: Understanding WSDL in a UDDI registry. Part 1, IBM (2001)
25. Siddiqui, B.: Using SOAP as a UDDI Search Engine (2001)

A Scalable Architecture for Real-Time Online Data Access

Ionuţ Roşoiu

University Politehnica of Bucharest
Computer Science and Engineering Department
Splaiul Independenţei nr. 313, Bucharest, Romania
ionut.rosoiu@gmail.com

Abstract. In recent years more and more computer users are getting connected to the Internet. The explosive growth not only provides a wealth of opportunities for building online services, but also poses significant challenges. Handling an ever increasing number of users imposes a careful design of the system. Giving them real-time access to the data complicates even further the implementation of such a service. In this paper a real-time data storage architecture is presented that scales with ease to accommodate an increasing number of clients. The proposed architecture can handle not only predominant read operations, such as when using a traditional database with a caching layer, but also when most of the operations are write operations. By also prioritizing the access to data with respect to the user interactions with the system, the real-time performance of the system is enhanced.

1 Introduction

Computers and the Internet have become an integral part of our life. Having one at home with an Internet connection is increasingly common for most of us. A few years ago, Mrs. Forrest, a teacher at the Taylorsville Elementary School, USA started a project with her sixth-grade students. She asked them to send a short email message to all their contacts asking them to forward it to their contacts and so on. They also asked each recipient to respond to the email so that they could keep a record of how many responded and how far the email has reached. In a matter of hours the email has reached Japan and had been responded to. Although the project was scheduled to run for a few months, because of the huge number of responses the project was cancelled only after a few weeks. In this short time it had already received more than 450.000 replies from eighty-four countries [16].

In this heavily interconnected world, online services face complex and difficult problems in order to accommodate their increasing user base. Designing such systems requires great expertise in order to allow them to scale cost-effectively. Many of them suffer greatly from an inappropriate use of databases. Although traditional relational database management systems offer advanced features, ACID properties and improved management functionality, they are also more expensive and often create congestion problems that turin into single points of failure for the whole architecture. Moreover, most of them are designed around the idea of vertical scalabilility, which although it's easier to implement, it is much more expensive, creates single points of failure and complicates replication and availability.

In order to service a high number of concurrent users and not increase the latency too much, multiple servers are used and data is replicated on each of them. This elegantly solves the problem if the majority of operations are reads, because no data has to be updated. However, when write operations have to be performed, all the replicated copies have to be updated at the same time, leading to a huge overhead and time penalties.

In practice, most of the advanced features offered by DBMSes are not actually needed. Maintaining ACID properties may not always be required, but only for certain operations or it might be completely bypassed. Recently, a whole bunch of non-relational data stores started to appear, designed from the ground up for horizontal scalability. Their creation was based on the observation that most of the online services today can be designed without using a complicated traditional database schema, such that the scalability of the system is increased.

In order to better understand some of the problems, we will present a real-time security protection system that enables users to safely use their Internet connection, whilst proactively protecting them against online threats.

Using personal computers doesn't come without security threats, such as phishing websites trying to steal confidential financial information, viruses that cripple the computer or worms that steal sensitive personal information or intellectual property. New threats emerge every day with more and more sophistication and an increasing destructive capability, some even powerful enough to infect nuclear facilities [13].

Most antivirus softwares rely upon a local database of virus signatures and sophisticated heuristics detection techniques in order to protect the users from both known and unknown malwares. Using a local database means that it has to be regularly updated and it may be out of sync with the newest threats for quite some time, leaving the unsuspecting user unprotected. This problem was also outlined in a report from Microsoft [3] and can be seen in Figure 1, where the reports about the Win32/Renos virus are displayed. It can be seen that the systems affected increased exponentially as time passed, affecting more and more computers. But once the virus signature was made available to the customers, the reports decreased dramatically. Had it been released earlier, a great number of infections would have not occurred at all.

An in-the-cloud system ensures that information about newly discovered, dangerous software is pushed to the users much faster. Also, because the service interacts with it's users in real-time, it can also deduce how the virus spreads around the world and act proactively to defend the non-infected users. Detecting the moment a virus outbreak starts is essential, as it allows the system to alert its users about the new threat more quickly.

The cloud system has to be composed of many servers around the world, not only to accommodate a large number of users, but also to decrease the latency of constantly communicating with them. However, the data exchanged with the users has to be aggregated in order to have a complete picture and new information about the threats discovered has to be pushed as fast as possible. What such a system would be required to do is to prioritize the distribution of new information according to how important it is to its users. It's obvious that detecting a new threat with a viral spread would be considered of maximum importance. Also, prioritization could be done based on the geographical location of the user. This is important because, as shown in Figure 2 from

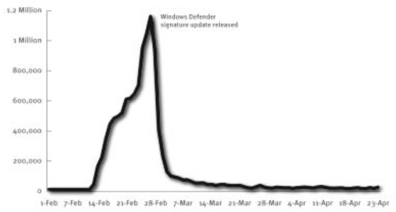


Fig. 1. Reports about the Win32/Renos virus affecting Microsoft Windows Vista

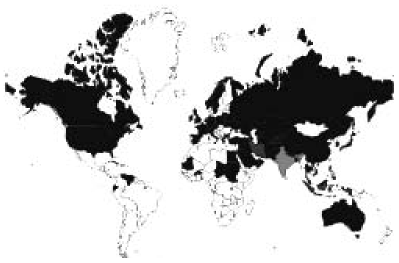


Fig. 2. Rootkit.Win32.Stuxnet world distribution (black is lower, gray is higher)

SecureList [5], some viruses spread based on this information. So, for example, a user in Iran would be first of all interested in receiving alerts about the Stuxnet virus spreading aggressively in his country, rather than a virus appearing in another corner of the world.

Storing information in the cloud using traditional DBMS in order to match these requirements would be very difficult, as this is not the usual *extract-transform-load* task [15] and data is constantly written to the system. Many such systems, such as MySQL [6] or PostgreSQL [7], have limited capabilities related to scalability. Sharding and replication are common techniques aimed at solving this problem. Apart from the problems they introduce themselves - performance degradation due to joins being made across different shards, referential integrity across different servers, difficult rebalancing - they don't solve the problem if the majority of operations are writes. Not only a real-time system that can handle queries is needed, but one that can handle mostly write operations. That is because, in order to detect outbreaks, the system constantly monitors the users and updates the information in the system based on their input, so that any abnormal behavior can be deduced and investigated.

It can be seen that maintaining all this data using relational databases poses a significant implementation overhead and doesn't completely solve the scalability issue. Moreover, the needed prioritization scheme has to be separately implemented, which leads to yet another difficulty in designing a proper database schema. The system proposed in this paper is well suited for such a task, as it scales horizontally with the number of nodes added and inherently prioritizes distribution of new data in the system according to the user's query patterns.

The layout of this paper is as follows. Section 2 outlines the relevant research in this field. Section 3 describes the architecture behind the described system. Some of the experimental results obtained are presented in Section 4. Finally, Section 5 provides concluding remarks.

2 Previous Work

One of the first non-relational database systems came from Google and is called BigTable [8]. Built upon other systems developed at Google, such as the Google File

System (GFS), it was designed to be highly scalable but also very fast. The data model is a three-dimensional table - a (row, column, timestamp) tuple - which holds an uninterpreted array of bytes. Row keys are specified by arbitrary strings and help the system store data according to the lexicographic order of the row keys. Distribution and load balancing works with tablets, that are formed by groups of rows. The system does not address problems such as frequent reconfiguration or Byzantine fault tolerance.

Dynamo [9] is a system developed at Amazon to reliably and efficiently solve the storage problem of key-value pairs. The system is eventually consistent and was designed like so in order to improve the scalability and availability requirements. Because availability was so important, the system will never reject an update, even if there is a problem with some nodes in the system. Data is split across the cluster using a consistent hashing scheme against the keys, in order to split them evenly between the nodes. The hash function's space is treated as a circular ring and each node is designated to store the data that corresponds to a portion in this ring. When an item is added to the system, it's key is hashed and based on the ouput of the hash function, the item is placed on the corresponding node. In order for the system to handle individual node failures, the same item is replicated on the next `N-1` nodes in the clockwise direction in the ring. Consistency is maintained between the nodes using a quorum-like protocol that has two configurable values: `R` - which is the number of successful read operations required when reading and `W` - the minimum number of successful write operations required when writing.

A distributed, column-oriented data store that was modeled after Google's BigTable is HBase [2]. Data rows are stored in labeled tables, with each row composed of a sortable key and an arbitrary number of columns. The table itself is stored sparsely, so that rows in the same table can have varying columns. Each of the columns can have multiple versions of the same key, making it easy to recover from bugs or detect write conflicts. The system has a tight connection to the Hadoop HDFS file system, which comes with a great disadvantage: logs are stored in the filesystem, but files don't exist until they are explicitly closed, leading to potential durability issues. It also has a single point of failure, the Hadoop NameNode, a problem that is scheduled to be fixed.

Another system, which has its roots at Facebook, is Cassandra [11]. The storage layer is placed atop of a P2P network. It combines features from both the Dynamo and BigTable systems and was designed to be highly available. In order to increase availability, consistency had to be dropped, so the data store is eventually consistent. However, the consistency level can be specified by the user, ranging from a ”writes never fail” model to a ”block all replicas to be readable” model. The system lacks atomicity guarantees for updates across multiple keys and relies upon a gossip protocol for cluster membership.

3 Design and Implementation

As outlined in the introduction, the goal of the service is to handle a high number of clients that perform *read* and *write* operations on the data. The data partitioning scheme used was first introduced by the Chord [14] system. Much like a distributed hash table, data is composed of key-value pairs stored on different servers in the system based on the hash value of the key. The database cluster is considered to be an M-bit integer space, divided into equally sized partitions, each handled by a different server. Keys are hashed and, based on the hash value, they are mapped to one of the partitions in the cluster. A node will typically be required to handle a number of keys equal to `number-of-keys / number-of-nodes`, just like in Figure 3. Exceptions to this rule happen when a new node enters the system or a node leaves it, an operation which imposes a reconfiguration phase in which keys are migrated between the remaining nodes, so that their number is again balanced across the ring.

The same address space is used to identify a node, each node having an unique ID in this address space. The node identifier is stored in this implementation using 34 bits, thus each identifier can have a value between 0 and 2^{34}-1. Node communications happens between adjacent nodes in the ring, i.e. each node can communicate with its predecessor and successor in the ring. The successor for a node is the node that has the lowest ID greater than the node's ID, in a clockwise direction following the ring. Conversely, the predecessor of the node is the first node in the counter-clockwise direction.

Just like in the Chord system, in order to minimize the number of hops needed for a message to reach its destination, each node maintains a routing table in which it holds its successors in the ring. When a message for another node arrives, each node inspects the routing table in order to forward the message to the closest node to the destination.

Each node in the cluster will store the keys (and their associated values) corresponding to a range in the domain of the hash function, meaning it will be the *master* for those keys. The node will be responsible with storing the keys that have their corresponding hash values between the predecessor's node ID and its own ID. Each value corresponding to a key can be read by any other node in the cluster, but only its master is allowed to change it.

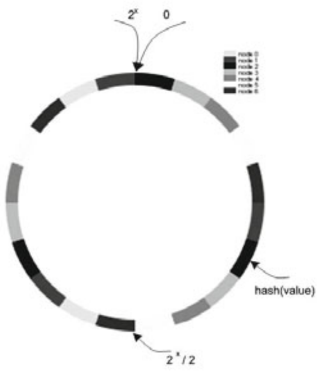


Fig. 3. The circular key ring is split between the nodes

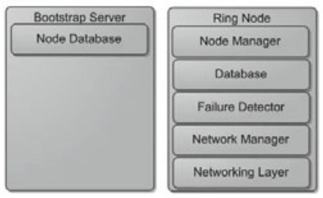


Fig. 4. A high-level view of the system

Because of the CAP theorem [10], which states that a system is capable of offering at the same time only two of consistency, availability and partition tolerance, a choice had to be made which ones were mostly needed. For this system, the least important was consistency, while the most important was availability. In order to service as many clients as possible the system is thus eventually-consistent, meaning that, at any given time, it may be in an inconsistent state. This actually means that the corresponding value for a key might be different on different nodes. It doesn't mean that the situation will last forever, it is just temporary.

Apart from this three characteristics, tolerance to failure had also been addressed. For it to be increased, each key is mapped not only to one, but more master servers, in this implementation four. Each key will be stored on the master node and on the next 3 successors after the main master node. The number of replicas for a key was not randomly chosen, but based on empirical observations. It is a number low enough to minimize the overhead of maintaining multiple copies of the same data, but high enough to actually help in increasing the failure tolerance. A number that is too big would have increased both the storage requirements as well as the network communication between nodes, while a number too small would have not helped in the process of error recovery from individual node failures. Maintaining several copies for the same key on different nodes gives the possibility to service requests for the key even if its primary master has died. When the key is written to, the corresponding value will not only be modified on a single node, but on all the four master nodes.

Because the system is not fully-consistent, the problem of conflict resolution had to be addressed. The system tries to automatically resolve the write conflicts by using Lamport's vector clocks [12]. When this is not enough to determine the merge outcome, a versioning scheme is used along a user-defined function that combines the different versions. Special care is taken in order to minimize the time the system is left in an inconsistent state. Another feature that is unique to this system is that it tries to achieve consistency faster for keys that are more "important" in the system, i.e. keys corresponding to information that will have the greatest impact to the users. Going back to the example in the introduction, whenever a new virus is detected and that virus has a viral spread, the detection information will be propagated with a higher priority than an information related to a less dangerous new virus.

The propagation of information based on data priority has a good motivation behind. It will impact positively a larger number of users that will benefit from the fact that they first receive important, real-time information. The number of users not affected by the consistency problem is thus lower for important data and higher for not-so-important data, so the system has a greater performance from the user's point of view.

A high-level overview of the system, with its main components, is given in Figure 4. The system has a *Networking Layer* that is responsible with handling the communications inside the ring and which maintains the routing table. The *Network Manager* is the subsystem that is in charge with handling the propagation with priority of the keys, being the middleware between the networking layer and the processing layer above. The *Failure Detector* is the component that is used to detect which nodes are not functioning properly, so that they can be gracefully removed from the ring. The main application logic is implemented in the *Node Manager* subsystem, while the *Bootstrap Server* is used to manage the IDs for each node in the system. The *Database* subsystem is in charge of storing the keys and their corresponding values.

The whole system was implemented using the ERLANG [1] language, which emphasizes message-based concurrency. Another advantage is that it's possible to use special processes, called supervisors, to monitor child processes and restart them whenever a failure is detected. Two different strategies can be used for a supervisor: the *one-for-one* strategy can be used to restart only a failed process within a group and the *one-for-all* strategy can be used to restart the whole process group whenever one of its processes dies. The *all-for-one* strategy was used to monitor the *Database*, *Failure Detector*, *Node Manager*, *Network Manager* and *Networking Layer* components.

Each node is assigned an unique identifier by contacting the *Bootstrap Server* before joining the ring. Then the node wanting to join will contact any other node in the ring, requesting the identifier of the node that will become its successor. It will then contact its successor to inform it that its predecessor has changed. After the predecessor-successor links have been updated, data is redistributed through the ring. It should be noted that even if the *Bootstrap Server* is the only centralized part in this infrastructure, thus suffering from the single point of failure problem, steps are being taken in order to remove it completely. Modifications inside the ring are detected by using a stabilization protocol that runs in the background. Each node periodically asks its successor which is it's current predecessor and, based on the response, it will decide if it should update its own successor or not. Also, each node will notify its successor about the fact that it's still alive, giving the successor a chance to also update its own predecessor. The successor will update its predecessor only if it knows no other node that it's closer to it, in a counter-clockwise direction on the ring. Whenever a node voluntarily whishes to leave the ring, it will will notify its successor and predecessor before disconnecting.

As noted earlier, in order to minimize the number of hops a message has to travel round the ring, a routing table is kept at the networking level inside each node. The routing table will have at most `log(M)` entries, with `M` being the maximum ID that a node can have. Each entry in the routing table contains a pointer to a node that follows the current node in the ring. Thus, the i^th entry at node *n* will point to the first node in a clockwise direction in the ring that is at least $2^(i-1)$ far apart from node *n*.

Because each main master node is responsible with storing the keys in the range `(predecessorID, nodeID]`, whenever a request for a key is received, the node first tests if the key hash falls between its own ID and the ID of its successor. If this is the case, the node will query its successor for the key. If not, the node will look-up in the routing table for a node that immediately precedes the searched node ID and forwards the request to that node. With this simple algorithm, the number of nodes that must be queried before the value is reached is `O(logN)` in a network with `N` nodes. The routing table is maintained by the networking layer by periodically sending a request to the successor. This procedure allows nodes to find out about the newcomers in the ring as well as allowing new nodes to create their own routing table.

One important aspect to note is that each node keeps a local cache, along the keys that it's responsible for. The cache is made up of keys that are not stored on the node, but have been queried by clients recently. Keeping such a cache allows the node to service requests for keys not stored on the node faster than synchronously querying the masters for that key. It also allows a greater number of clients to be serviced by each node. Whenever a request for a key that it's not in the node or in the cache arrives, the node will immediately return a `null` value to the client, because it doesn't yet know what the value for the key is, like in Figure 5. It will then asynchronously query one of the masters for that key to find out the corresponding value and it will update the cache accordingly. This way, when a new request for the same key arrives it will service it from the cache. If the read request comes for a key for which the node is a master, the node will service it using the local value, but from time to time it will also asynchronously check with the other masters if the key has been updated in the meantime.

When a key is written, the value is either updated on the node and the write is forwarded to the other three masters for the key (if the node is the master for that key) or the cache is updated and the write is forwarded to the other four masters (if the node isn't the master for that key).

The system is capable of automatically assigning priorities to keys based on the interactions with the clients. A user defined function that computes the priority can also be specified. The automatic priority assignment works by monitoring how many *read* requests come in for each key. Whenever e key that is stored on the node is *read*, its corresponding priority is increased by the *Network Manager*. If the same key is *written* to, the new value will have to be sent to the other master nodes, but keys will be sent according to their priority. Using this simple scheme, the system ensures that the time in which an oftenly-used key is not consistent is kept to a minimum. For such *hot* keys, more clients will not be affected by the consistency problem, thus improving the overall system performance from the user perspective. If the key that is written is a key for which the node is a master, the new value just replaces the one currently stored. The node also proactively informs the other three masters about this write, also taking into consideration the priority for the key as shown in Figure 6. This procedure is performed to aid in achieving faster consistency for *hot* keys.

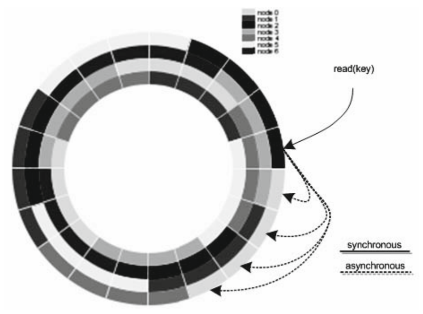


Fig. 5. How read operations are handled

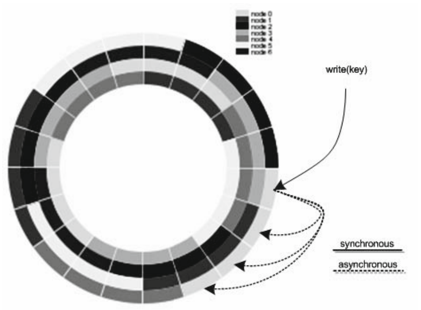


Fig. 6. How write operations are handled

4 Results

This section presents some of the results obtained while testing the system. For the tests, two computers were used with the specifications in Table 1 for simulating the servers and five computers with the specifications in Table 2 for the client machines.

Table 1. The server computers characteristics

```
model name      : Intel(R) Core(TM) CPU 750 @ 2.67GHz
cpu MHz         : 2667.263
cache size      : 8192 KB
bogomips        : 5333.24
memory          : 6GB
```

Table 2. The client computers characteristics

```
model name      : Intel(R) Core(TM)2 CPU 600 @ 2.40GHz
cpu MHz         : 2394.000
cache size      : 4096 KB
bogomips        : 4616.46
memory          : 1GB
```

The first test was with a larger number of clients. 2700 clients on five computers and 6 servers on two computers were simulated, with each client running 1000 iterations consisting of two read requests and two write requests, thus the read-write ratio was 50%-50%. Each request was made to a different server, chosen in a round-robin fashion. The results shown in Figure 7 show an average of 4700 requests per second for each machine and an average of about 23000 requests for the whole system. By monitoring the cluster with the MonaLISA [4] distributed monitoring system it was observed that the number of simulated clients was too big considering the limited infrastructure available. Swapping to disk because of not enough RAM memory available was the biggest bottleneck while conducting the test.

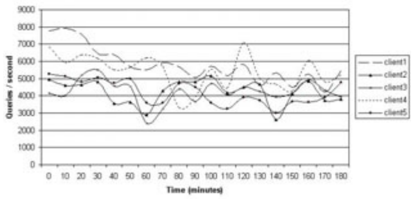


Fig. 7. Request rate for a split read/write test with 2700 clients

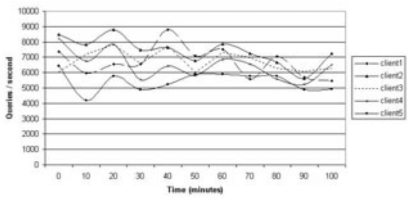


Fig. 8. Request rate for a split read/write test with 1200 clients

The second test used a smaller number of simulated clients. 1200 clients making the same 1000 iterations with a split 50% reads - 50% write ratio were used. The results obtained in Figure 8 show an increase in the throughput per machine to about 6500 requests per second, while the total number of queries serviced by the system gravitates around 32000 queries per second. The same big overhead per machine was the motivation that lead to a further decrease in the number of simulated clients.

The next test used only 960 clients split evenly on 5 computers, with 192 clients simulated per machine. Each client performed 1000 iterations with a read-write ratio of 50%-50%. The results shown in Figure 9 show an average of 7200 requests/second per machine, with the total throughput of the servers at 36000 requests per second. The request rate for a random client on each of the five client machines was also measured and the results are shown in Figure 10. The average request rate is about 36 requests/second for the sample selected population.

For the same number of clients, a simulation that took about 90 minutes was done, but with a different read-write ratio, of 25% reads - 75% writes. Each client performed 1000 iterations with four operations per iteration. The first operation was a random key

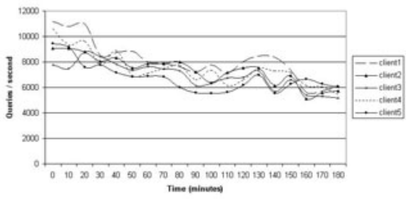


Fig. 9. Request rate for a split read/write test with 960 clients

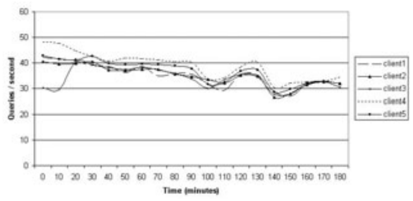


Fig. 10. The requests rate for one of the 960 clients on each machine

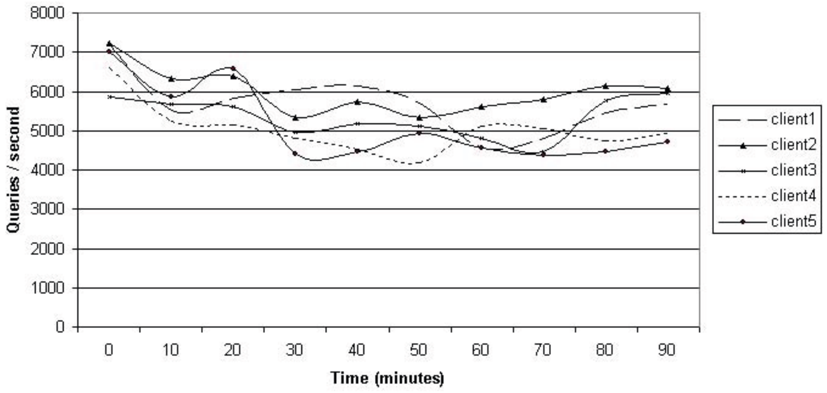


Fig. 11. Request rate for a 25% read, 75% write scenario with 960 clients

read, while the other three operations were random key writes. The results presented in Figure 11 show an average of 5500 requests/second per machine, while the total throughput of the system is at 27000 requests per second. It can be seen that if the number of writes is higher, the total request rate drops with about 25%, because there is a higher cost of achieving consistency when the majority of the keys are changed often. Each master node will have to inform all the other three master nodes about each key that is written, thus increasing the communication between the servers. Because the writes are done randomly by each client, a write request will not necessarily reach the master server first, so the request will have to be propagated through the ring, which also increases the communication overhead inside the ring. Taking into consideration all these additional overheads, the the request rate achieved seems more than reasonable.

5 Conclusions

The system described in this paper can mediate access to real-time data in a horizontally scalable fashion. It automatically ensures that the data that is most used in the system will be made available to all of the clients with increased priority. Because the system is an eventually consistent one, this consistency issue is solved taking into account how important the data is to the clients. From the user's point of view, the system behaves more efficiently, because heavily used data will be up-to-date much faster. Achieving consistency in a distributed environment taking into consideration the actual usage patterns of the clients is a new idea, which has a low impact in the performance of the system expressed in queries per second, but a major positive one from the client's perspective.

However, much more work remains to be done. A further increase in the total throughput of the system is currently being investigated, as well as assessing the impact of frequent ring reconfiguration on the performance of the system. The test results presented in this paper were obtained using a rather limited test infrastructure, with computers not too well suited for the job because of their hardware configuration. The future plan is to test the system using more powerful computers and in a greater number. Future test plans also include the benchmark of the latency in achieving consistency for keys that have great importance compared to the ones that are less important.

References

[1] The ERLANG programming language, `http://www.erlang.org/`
[2] Hadoop database, `http://hadoop.apache.org/hbase/`
[3] Microsoft Security Intelligence Report - volume 7, `http://download.microsoft.com/download/A/3/0/A30A60D9-1303-4B6A-91B7-BB24E0211B05/Microsoft_Security_Intelligence_Report_volume_7_Key_Findings_Summary_English.pdf`
[4] MONitoring Agents using a Large Integrated Services Architecture, `http://monalisa.caltech.edu/monalisa.htm`
[5] Myrtus and Guava, episode 3, `http://www.securelist.com/en/blog/272/Myrtus_and_Guava_Episode_3`
[6] The MySQL database, `http://www.mysql.com/`
[7] The PostgreSQL database, `http://www.postgresql.org/`
[8] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. In: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2006, Berkeley, CA, USA, pp. 15–15. USENIX Association (2006)
[9] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: Amazon's highly available key-value store. SIGOPS Oper. Syst. Rev. 41(6), 205–220 (2007)
[10] Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33(2), 51–59 (2002)
[11] Lakshman, A., Malik, P.: Cassandra: structured storage system on a P2P network. In: Proceedings of the 28th ACM Symposium on Principles of Distributed Computing, PODC 2009, pp. 5–5. ACM, New York (2009)
[12] Lamport, L.: Time, clocks, and the ordering of events in a distributed system. ACM Commun. 21(7), 558–565 (1978)
[13] Marks, P.: Stuxnet: the new face of war. The New Scientist 208(2781), 26–27 (2010)
[14] Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications, pp. 149–160 (2001)
[15] Stonebraker, M., Abadi, D., DeWitt, D.J., Madden, S., Paulson, E., Pavlo, A., Rasin, A.: Mapreduce and parallel dbmss: friends or foes? ACM Commun. 53(1), 64–71 (2010)
[16] Watts, D.J.: Six degrees: the science of a connected age. Norton, New York (2004)

Socially Responsive Resource Usage: A Protocol

Hrushikesha Mohanty

Department of Computer and Information Sciences
University of Hyderabad
India
mohanty.hcu@gmail.com

Abstract. Sharing of state resources like natural resources and developmental funds, needs to be inclusive as well as traceable. Further, for sustainability resource rights are to be enforced. For the purpose, a protocol for resource usages is proposed and shown that the protocol could be made adaptive to empower people by cooperation, collaboration and self help.

Keywords: Social Distance, Protocol, Social Inclusion, Resource Rights for Sustainability.

1 Introduction

Resources empower an individual to sustain and excel. Resources are either natural or man-made. Here, we are mainly concerned of fair distribution of public resources like state funds and natural resources that are usually used for poverty elevation, personal uses and business purposes. This paper proposes a resource usage protocol with prime aim of protecting both user and resource rights. User rights include fair chance of resource usage whereas resource rights include protection and conservation of resources. The proposed protocol presents a framework for communication among users and resources (i.e agencies managing resources).

The proposed protocol is useful for orderly usages of natural resources and public resources like coal and house building aid respectively. In general these resources are managed by designated agencies; a user interacts with it for availing a resource e.g. a saw mill owner wanting logs approaches forest department for permission. Another example could be of state resources like house building aid available to underprivileged. The novelty of the proposed protocol includes:

- a generic framework for public resource management
- assurance of fair order to access resources promoting social inclusion
- ensuring protection of resource rights
- traceability of resource usages

The paper has another six sections. The second section presents social ethics on public resource usages. In third section, the entities and their relations used in design of the protocol are specified in class diagrams.In the fourth section, the protocol stack and its behaviour are modeled. An analysis of the dynamic behaviour of the protocol is carried out in the fifth section. Some related works are surveyed in the sixth section. Seventh section concludes the paper.

2 Ethics on Resource Usage

The synthesis of the proposed protocol is based upon two principal ethics that are, *socially responsive resource usage* and *resource rights for its protection.* A user or a group for survival or growth looks for resources; a request for resources is to be honoured based on social norms. A socially disadvantaged person needs resources mainly for survival whereas a user with knowledge and capital needs resources for commercial usages; thus making profit for personal growth. Rights and privileges are to be associated with constraints particularly when natural and public resources are in use. While unlimited usages of resources is to be banned, users should be made responsible to maintain the resource in proportion to its use. At present situation in prospect of environment, a user on using a natural resource should pay in certain proportion of use, for safe keeping of the ecosystem.

A resource usage is viewed as a process that starts from initiation of resource request to fulfilling of obligation for resource usage before the process terminates. That way, a resource usage session has a life time spanning from initiation to termination. During the session, both user privileges and resource rights are guarded by checking the satisfiability of constraints and obligations. Resource constraints follow *just enough* principle to safeguard resources from unabated exploitation. It also follows *just fair* principle that prioritized under-privileged users based on their *social distance*s. Social distance is a notion that quantifies how far a person is from a resource. More the distance, the more the person is at disadvantage to assess the resource. It is to note that social distance is coined to bring in social issues in synthesis of the proposed protocol. A resource user is obliged to replenish, to recreate or to maintain the resource it used. This ensures the resources' rights to protect itself for humanity. These obligations are made by legislation's and social practices. The constraints defined on resource usage are useful to constrain over-usages or misuses of resources like indiscriminate mining, deforestation or squandering of poverty elevation funds.

The proposed protocol has a social objective to empower people those are at disadvantageous position in accessing public resources. The protocol prioritizes requests of the socially distanced people for resource sharing. A logical distance defined between a resource and its user, is termed as social distance of the user with respect to the resource. A resource has three attributes: *place*, *cost* and *constraints*, having dominating effects on its usages. A place means the location where a resource is available or from where the resource can be availed. A resource may have specified cost that a user has to pay for availing it. The constraints are set of conditions and assertions used as pre- and post-conditions to a resource usage. Let's take HBF (House Building Fund), a resource available for marginalized people and the resource is specified as:

```
Resource::r
     {
         Location: Welfare Dept., Bhubaneshwar,Odisha
         Cost: 1000 Rs.
         Constraint: AnnlIncomeL4() &  FamilyDependent4()  &
                                 CompletionCert()
       }
```

The resource HBF is being distributed by Welfare department. Bhubaneshwar, Odisha (gives a geographical location of the resource); the cost to apply for fund is 1000 rs. The fund is available for those who have less than four lakh rupees annual income *AnnlIncomeL4()* and the dependents are four or more *FamilyDependent4()* . These two are *pre-conditions* to avail the fund. On availing the fund, a user needs to submit a house completion certificate*CompletionCert()*. This is a post-condition to a usage of the resource. One can add assertions, invariants related to governance, environment, community usages and any other relevant domains. These are to be verified during a resource usage session or after. Preconditions can be verified with the facts available at a resource requesting user.In order to make the resource accessible to socially disadvantaged users preconditions could be made friendly to them whereas, *postconditions* are for resources to assert their rights. This brings out the inherent dichotomy between liberal development and strain on resources. We have handled this issue making the protocol adaptive as shown later in subsequent section. A user alike a resource is specified as:

```
User:: u
     {
       Location: Bangalpur, Balasore, Odisha
       Invest: 1500 Rs.
       Fact: (Dependents 5) & (AnnlIncome 3L)
     }
```

A user is specified by its geographic location, the amount it can invest and some facts as presented of types viz. Location, Invest and Fact, respectively. The geographical distance between user and a resource is calculated from *r.Location* and *u.Location*. And financial distance between two is computed from *r.Cost* and *u.Invest*. The conditions *AnnlIncomeL4()* and *FamilyDependent4()* are evaluated with *u.Fact*.

Social distance between a person and a resource is computed by

$$SD^r_u = w_1 * LD^r_u + w_2 * ID^r_u + w_3 * CD^r_u \tag{1}$$

where LD^r_u, ID^r_u and CD^r_u correspond to location distance, invest distance and condition distance respectively for resource r and user u. More, w_is are the weights assigned to each of the three constituents of the social distance metric. LD^r_u computes absolute geographical distance between a resource and a user. Based on the distance, it is categorized as far, near or at average distance. And each of it assumes value in 10 scale e,g, 10 for *far*, four for *near* and six for *average distance*. Similarly $ID_u^r = r.Cost - u.Invest$. This factor contributes to social distance only if $ID_u^r > 0$. Semantically, $ID_u^r > 0$ is categorized *prohibitive*, *expensive* and *costly* based on slabs defined on investment distance; let's say the three assume values ten, eight and six respectively in a scale of 10. Of course, this categorization has to be person specific for example based on one's economic conditions and liability, the ID slabs are to be fixed. The conditional distance indicates a person's social limitations in availing a resource. A person is disadvantaged if $u.Fact$ satisfy the conditions stated at $r.Constraint$. If no condition is satisfied then CD_u^r is zero. Please note that, we assume constraints with a resources are framed in favour of disadvantaged. When say $x^{'}$ conditions are satisfied from total x conditions then $CD_u^r = (x^{'}/x) * 10$; its different ranges of values say 10-8, 7-5, 4-1 are identified as the most disadvantaged, more disadvantaged and disadvantaged respectively and they numerically contribute to the metric 10, 8 and 6 respectively. In order to express SD_u^r in 10-scale, we propose weight values in the range of (0,1) based on the importance we lay to each component. For the sake of discussion let's assume all three are of equal importance and so each assumes value 0.33.

Other than being able to define social distance for a person from a resource we plan to introduce resource rights in $r.Constraint$ in terms of *Pre* and *Post* conditions. In precondition a resource specifies its choices to serve. Here we make a resource more reachable for socially distanced people by prioritizing their resource requests. Also preconditions can be set in such way that the affluents get less privileged access to the resource. Thus a resource usage becomes *socially responsive*. Further, postcondition specifies the obligations of users that should be fulfilled on utilization of resources. We can have assertions and in-variants for ensuring legal resource usages.

In the design of the proposed protocol the social ethics taken into consideration are:

1. Socially underprivileged should have better chance to avail a sharable public resource.
2. Financial and social conditions of a person have impact on its ability to avail a resource. Further, a remotely located person is found at disadvantage in availing state resources.
3. A resource has means to exercise its right to protect itself and grow.

In the next section we introduce the important artifacts that make a framework for implementing the proposed protocol.

3 Participating Entities

Prior to detailing of the proposed protocol, here we will introduce the entities' that participate in the design of the proposed protocol Fig. 1. Each user registers at a *user router*: *ur*, which collects resource requests from users and keeps track of the delivery of resources to users. In reality resource delivery can be in both

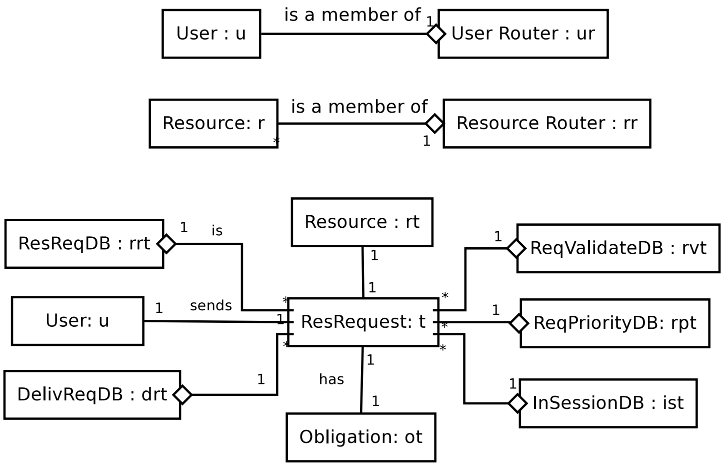


Fig. 1. Participating Entities and Relations

ways i.e. off-line/on-line. But in both the cases the delivery details are maintained at user router. Similarly, the roles of a *resource router*: rr can be thought of. A resource router can be imagined as an agency that allows, monitors and controls usages of resources. Each resource registers at a rr that executes the protocol for resource usage like a ur. That is, the copies of the proposed protocol stack $Fig.2$ resides at ur and rr for ensuring socially responsive resource usages. A resource request $ResRequest$ originates at a user and gets forwarded to a resource owner through respective ur and rr. So, for each $ResRequest$, there exists a user u, a resource rt and corresponding ur and rr. Each $ResRequest$ has a an obligation ot of $Obligation$ based on the resource it looks for. The entity obligation specifies what a user needs to oblige for using the resource it's looking for. At different stages during execution of a resource request, information relating to the request processing are recorded at different databases viz. $ResReqDB$, $ReqValidateDB$, $ReqPriorityDB$ and $DelivReqDB$. The use of these databases are explicit in Algorithm 1.

4 The Protocol

A user and a resource (owner/mentor) exchange messages between them during a resource usage session. Resource usage is a process initiated by a user and in between resource and user both the routers ur and rr take part in message communication. Both the routers validate, prioritize, route and record messages transacted in resource usage sessions as shown in *Algorithm* 1, SRRUP: Socially Responsive Resource Usage Protocol.

Algorithm 1. Socially Responsive Resource Usage Protocol

```
t: transaction for a resource request.
ResReqDB: Resource request database containing transactions.
rt: a resource requested by a transaction.
ReqValidateDB: Database of validated requests.
rvt: a valid request in ReqValidateDB
ReqPriorityDB: Database of validated and prioritized requests.
rpt: a prioritized request in ReqPriorityDB
InSessionDB: Database of running transactions.
ist: a transaction listed InSessionDB
DelivReqDB: Database of transactions that received resource.
drt: a transaction in DelivReqDB
Obligation: Condition resource user to meet on availing requested resource.
ot: an obligation with respect to a transaction.

// On taking a request from ResReqDB do at ur
for ( t ϵ ResReqDB) do
   if !Validate(t) then
      Terminate(t)
   else
      PutInReqValidateDB(t,ReqValidateDB
      Prioritize(t,ReqPriorityDB)
   end if
end for

// On taking a request from ReqPriorityDB do at ur
for (rpt ϵ ReqPriorityDB) do
   PutInSessionDB(rpt,InSessionDB)
   SendResReq(t,rr) // send request to rr
end for

// On receiving resource request t from ur do at rr
On receiving SendResReq(t,rr)
PutInSessionDB(t,InSessionDB)
if !Validate(t) then
   Terminate(t) // Terminates the request
   SendTerminate(t,ur) //Informs ur to terminate.
   PutInReqValidateDB(t,ReqValidateDB
else
   PutInReqValidateDB(t,ReqValidateDB
   Prioritize(t,ReqPriorityDB
   SendDelivReq(t,r) // rr asks resource owner to deliver the resource to user
   //Resource delivery could be online or offline
   SendDelivReq(t,ur) // rr informs resource grant to the requesting user
end if

//On receiving SendDelivReq(t,ur) from rr processed at ur
sendResOblige(t, u) //Informs user to oblige for the resource grant
// User sends SendOblige(t,ur,rr) to ur and rr on obligation compliance
On receiving SendOblige(t,ur,rr)
// Executed both at ur and rr
if  Validate(t) then
   //Transaction obligation is validated
   PutInSessionDB(t,InSessionDB)
   Terminate(t)
end if
```

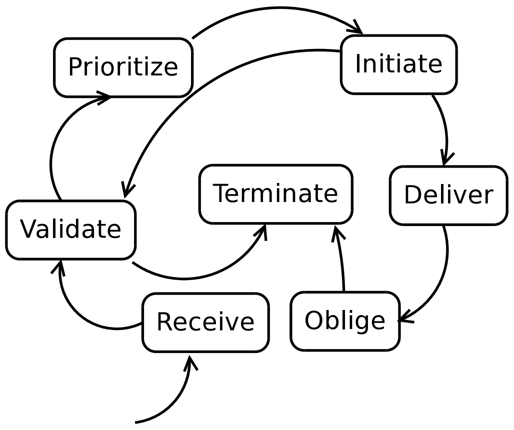


Fig. 2. Session State Machine

A *ur* at a time or periodically or asynchronously collects a *ResRequest* and validates for onward processing else, the request is rejected and the user is informed of the termination of the session. A user can be prohibited to use a resource either based on own social status or on performance history (e.g. a well off person say more than 4Lakhs rupees annual income is not allowed to avail house building aid from government, a defaulter to oblige resource rights like not paying to environment conservation fund after mining). For this at user router, resource rights and users obligation compliance information should be published; so that a request can be validated (a defaulter's request is terminated). The validated requests are prioritized based on users social distances and routed to a resource router *rr* corresponding to the resource. For that, *ur* needs to have resource addresses. For each request *ur*, creates a session and logs the session related activities in different databases till the session terminates. A session on resource usage terminates for one of the reasons: on rejection, on resource delivery and meeting resource obligation by the concerned user. Unless the user satisfies obligatory conditions, the session remains active and used for filtering out further requests if any from the same user. This information, can be further used by a related legal agency for further actions. So, the sessions as we see are *long-lived transactions*. A resource router *rr* on receiving a request performs the same operations as a *ur* does. It prioritizes the requests received from different user routers and issues the resources to each eligible requests in turn. This delivery status is also intimated to the corresponding *ur* for record as well as passing information to the corresponding user. A user on receiving the desired resource should oblige to specified resource usage obligations. And compliance to such obligation is recorded both at *rr* and *ur* associated to a transaction. And then the protocol terminates and the status is recorded at both the routers. We assume no loss in communication to avoid session status inconsistencies i.e missing of information at databases located at both the routers. The states a session assumes are *Receive*, *Validate*, *Prioritize*, *Initiate*, *Deliver*, *Oblige* and

Terminate. The state machine in Fig. 2 models the sequence of state changes in a session due to the protocol. The protocol is detailed in *Algorithm* 1 in terms of messages and associated actions at different routers.

The *Algorithm.* 1 provides an algorithmic implementation of the protocol stack Fig. 3. The algorithm is made self-explanatory by using meaningful function names. For the space constraint we refrain from formal analysis of the protocol. In the next section we extend the protocol to be adaptive resulting social inclusion.

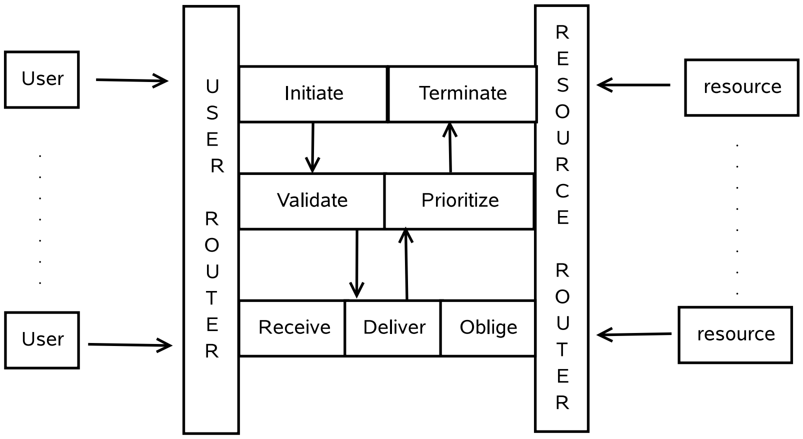


Fig. 3. Protocol Stack

5 Adaptive Protocol

As said earlier, the protocol is inclusive of have-nots as they score more satisfying the conditions that are usually coded in favor of them e.g. a user below poverty line has higher priority to get aid for house building. But, usually these people are ill-capable of meeting resource obligations like paying for resource investment or to fund for resource maintenance and generations. This brings out a trade off between inclusiveness for poverty eradication and resource sustainability. In order to meet this dichotomy, we propose to make the protocol adaptive so that both socially distanced people as well as resource will have win-win situation. For the purpose, three schemes viz. *Cooperate*,*Collaborate* and *Empower* are proposed and elaborated as follows:

Case-1: Cooperate A disadvantaged user while satisfying resource preconditions that is being a candidate for social inclusion, could be constrained to meet the cost to invest or for resource maintenance. Overlooking of this requirement of a resource, could have impact in long run. The protocol at this stage can look for cooperation from other users to overcome the constraint (e.g. by lending the resource cost). The cooperation among two users looking for the same resource is defined as:

$Cooperate(u, u^{'}, r) :: \equiv$
iff $(u.Invest - r.Cost) \geq (r.Cost - u^{'}.Invest)$
$\wedge AgreeToLend(u, u^{'}, r)$.
A user u cooperates with another user $u^{'}$ for accessing resource r provided it has invest amount extra after meeting its own requirement i.e for meeting the cost for resource r, so that it can lend the invest cost to $u^{'}$. The cooperation taken place only after the user u agrees. The process of cooperation is run at ur before starting the resource request session for $u^{'}$ starts.
Case-2: Collaborate. Two users u and $u^{'}$ can collaborate for resource r when individually they can't satisfy its post conditions and cost for availing the resource. But together they can meet, hence they cooperate. A cooperation can include more than two users. In a cooperation the constituent members contribute to acquire a resource that they may like to share (e.g. resource to set up a co-operative business). Collaboration of users creates a virtual user $u^{''}$ to access the resource. The process of collaboration that takes place at ur first creates a collaborator as:

$CreateCollaborate(u, u^{'}, r) \longrightarrow$
$(u^{''} \mid u^{''}.Location = mindist(u.Location, u^{'}.Location, r)$,
$u^{''}.Invest = u.Invest + u^{'}.Invest$,
$u^{''}..Fact = u.Fact \sqcup u^{'}.Fact)$.
The collaboration between u and $u^{'}$ for resource r creates a virtual user $u^{''}$ that assumes location equals to the minimum of the distances from u and $u^{'}$ have to reach resource r. Again the sum of their investments makes the investment of $u^{''}$ and so also for the Facts. Now,

$Collaborate(u, u^{'}, r) ::\equiv$
iff$(u.Invest < r.Cost) \wedge (u^{'}.Invest < r.Cost)$
$\wedge\ (u^{''}.Invest \geq r.Cost)$
$\wedge\ \neg SatisfyPostCond(r.Constraint, u.Fact)$
$\wedge\ \neg SatisfyPostCond(r.Constraint, u^{'}.Fact)$
$\wedge\ SatisfyPostCond(r.Constraint, u^{''}.Fact)$
$\wedge\ AgreeToCollaborate(u, u^{'}, r)$. // both agree to collaborate
Case-3: Empower. A user lacking capability to invest for and to satisfy post conditions of a resource r, may look for another resource $r^{'}$ for which he is capable to access. And the user is proposed by ur to avail the resource $r^{'}$. This is a kind of prescriptive approach a user router ur employs to empower a user u and possibly *empowering* it to avail its originally intended resource r. This is a process of engagement and empowerment leading to social inclusion without giving a scope for frustration. The process of empowerment of u that takes place at ur, by offering resource $r^{'}$ in place of r is:

$Empower(u, r, r') ::\equiv$
iff$(u.Invest < r.Cost) \wedge (u.Invest \geq r'.Cost)$
$\wedge\ SatisfyPostCond(r'.Constraint, u.Fact)$
$\wedge\ AgreeToTransact(u, r, r')$.
In reality, this prescription is made from experience of an agency, to empower an under-privileged user. For example, such a prescription could be of a government funded program that assures returns. Say here, on availing the resource r' the user has improved on capability required to access the resource r i.e. satisfies $(u.Invest \geq r.Cost) \wedge SatisfyPostCond(r.Constraint, u.Fact)$. Though, we have shown the empowerment of u for r in one step; still it can be extended to several levels of empowerment by just repeating the process considering several transactions to perform in sequence or in parallel or in mix mode even.

6 Related Work

In developing countries like India, poverty eradication program is mainly prescriptive and follow top-down approach. However, the current trend has taken bottom-up approach. Well-informed individuals and self-help groups have started taking decisive roles for self-development and poverty eradication. These entities mostly seek access to natural and public resources to carry out their plans and programs. Every citizen has right of access to state resources and at the same time each resource primarily natural resources also have right to protect themselves from misuse as well as overuse [9]. This paper proposes implementing resource rights (constraints) for access, recovery and regeneration. A resource is described in a way so that its social roles and responsibilities can be specified and queried [10]. A generic meta-model for representation of resources that are used in workflows is presented in [11][12]. The proposed model has provision to assign static meta model e.g. relation types, access policies and dynamic meta model to monitor and synchronize resource access to workflow activities.

The proposed protocol uses a concept of *social distance* for routing of resource requests. After designing the protocol, I found the term is in use since long mainly among social scientists. A way back, in 1959 Edward T Hall while describing people's world view, outlined how culture control people's lives and make one different from another. He introduced proxemics and use of space in his seminal works *The Silent Language* and *The Hidden Dimension.* He proposed several distances viz. Intimate Distance, Personal Distance, Public Distance and Social Distance. He said, social distance is the distance for impersonal business that shows the degree on involvement and formality.[13][14]

In [15], the research shows that self-enforcing exchange among socially heterogeneous agents i.e. socially distant agents can even work by adopting degrees of homogeneity with another, thus reducing social distance and making the arrangement workable. A generalization as well as extension to social distance is proposed in [16] and a multiagent frame to simulate the model is proposed.

In this paper we plan to make use of social distance for developing a protocol for sharing resources mainly natural and state funded ones. Information systems have been successful for societal uses by managing huge information and providing services like facility reservation systems, governance systems etc. But, there are very few works that model social phenomena and prescribe socially responsive solutions to the problems that the contemporary society is passing through. We have not come across much work on this aspect in literature. Some of the work we feel related are reviewed here.

In [4] a method to model social features like group, role, actions, dependency and interactions is demonstrated by extending UML. The proposed framework models concepts existing in organizational models for agents, including AALAADIN [7], dependency theory [6], interaction protocols [8] and holonics [5]. Negotiation among a group of ISPs (Internet Service Providers) has been proposed in [17] where the services are forwarded on a path negotiated by associated ISPs so that all will have win-win situation with respect to their business goals. A worldwide concerns on food supply chain include sustainability as well as transparency. Sustainability addresses wider global issues like environmental as well as social issues. A recent work [18] surveys on information systems that provide sustainability as well as transparency among stakeholders. Following the trend, it's found that one school works for introducing social concerns in development of information systems i.e. making it socially responsive while the other school works to make use of social concepts in development of systems for better performance. The work reported in this paper falls into former category.

7 Conclusion

Social distance, a social phenomenon, is mainly conceptualized to categorize social communications. This paper has redefined the term for calculating one's distance from a given resource and has devised a protocol for resource access based on this distance. The protocol is designed to address today's social concern like resource sustainability, transparency in resource usages and achieving social inclusion providing priority to socially deprived people.

The proposed framework with user router, resource router and protocol stack shows the possible implementation of the protocol. The working of the protocol stack is discussed by a state machine that presents the life cycle of a transaction. And the actions due to the state machine is presented in *Algorithm* 1. Further, the protocol is made adaptive incorporating social behaviours viz. cooperation, collaboration and empowerment for achieving social inclusions.

The immediate future work in sequel to this, is the implementation of the protocol and finding its usability in a given domain. A formal study on social distance and the protocol is planned to analyze several properties of the protocol. Of course, modeling social distance itself requires a very insightful investigation on dynamics of a particular society. Still, we view the concept of social distance has significant potential in the development of socially responsive information systems.

Acknowledgements. This work is carried out while I'm in sabbatical from University of Hyderabad. I would like to thank Tomasz Janowski, UNU/IIST and other anonymous reviewers for their valuable comments.

References

1. Kossinets, G., Kleinberg, J., Wattsí, D.: The structure of information pathways in a social communication network. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 435–443. ACM, New York (2008)
2. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. ACM Comm. 21(7), 558–565 (1978)
3. Adar, E., Zhang, L., Adamic, L.A., Lukose, R.M.: Implicit structure and the dynamics of blogspace. In: Workshop on the Weblogging Ecosystem (2004)
4. Van Dyke Parunak, H., Odell, J.J.: Representing social structures in UML. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 1–16. Springer, Heidelberg (2002)
5. Fischer, K.: Agent-based design of holonic manufacturing systems. Robotics and Autonomous Systems 27(1-2), 3–13 (1999)
6. Castelfranchi, C.: Founding Agent's 'Autonomy' on Dependence Theory. In: Proceedings of 14th European Conference on Artificial Intelligence, pp. 353–357. IOS Press, Amsterdam (2000)
7. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: Proceedings of Third International Conference on Multi-Agent Systems (ICMAS 1998), pp. 128–135. IEEE Computer Society, Los Alamitos (1998)
8. Odell, J., Parunak, H.V.D., Bauer, B.: Representing Agent Interaction Protocols in UML. In: Proceedings of Agent-Oriented Software Engineering, pp. 121–140 (2000)
9. Murali, K.S.: Microfinance, social capital and natural resource management systems: conceptual issues and empirical evidences. Int. J. Agricultural Resources Governance and Ecology 5(4), 327–337 (2006)
10. Nickles, M., Weiss, G.: A Framework for the Social Description of Resources in Open Environments. In: Klusch, M., Zhang, S.-W., Ossowski, S., Laamanen, H. (eds.) CIA 2003. LNCS (LNAI), vol. 2782, pp. 206–221. Springer, Heidelberg (2003)
11. zur Muehlen, M.: Resource Modeling in Workflow Applications. Issues and Perspectives Information Technology and Management 5, 271–279 (2004)
12. Xiao, Z., Chang, H., Wen, S., Yi, Y., Inoue, A.: An Extended Meta Model for Workflow Resource Model. In: Lang, J., Lin, F., Wang, J. (eds.) KSEM 2006. LNCS (LNAI), vol. 4092, pp. 525–534. Springer, Heidelberg (2006)
13. Hall, E.T.: The Hidden Dimension. Doubleday, GordonCity (1996)
14. Hall, E.T.: The Silent Language. Anchor Books, NY (1990)
15. Leeson, P.T.: Social Distance and Self-Enforcing Exchange. Journal of Legal Studies 37, 161–188 (2008)
16. Wąs, J.: Multi-agent frame of social distances model. In: Umeo, H., Morishita, S., Nishinari, K., Komatsuzaki, T., Bandini, S. (eds.) ACRI 2008. LNCS, vol. 5191, pp. 567–570. Springer, Heidelberg (2008)
17. Mahajan, R., Wetherall, D., Anderson, T.: Negotiation-Based Routing Between Neighboring ISPs. In: USENIX Proc. NSDI, pp. 29–42 (2005)
18. Wognum, P.M(Nel), Bremmers, H., Trienekens, J.H., van der Vorstb, J.G.A.J., Bloemhof, J.M.: Systems for sustainability and transparency of food supply chains Current state and challenges. Advanced Engineering Informatics (2010) (Article in Press)

An Automated HSV Based Text Tracking System from Complex Color Video

C. Misra and P.K. Swain

School of Computer Application
KIIT University, Bhubaneswar-751024, India
cmisra@yahoo.com, prasantkiit@gmail.com

Abstract. Tracking, extraction and recognition of text from video are important steps in building efficient indexing and retrieval systems for multimedia databases. We propose a top-down approach in which multiple cues are used for detecting text blocks in videos. In our proposed text tracking and extraction system, color and edge features are combined together to increase the precision of text detection. Our system detects both caption text as well as scene text of different font, size, color and intensity. Our objective is to detect both stationary text and moving text from complex color video. Extracted texts from videos are recognized using an OCR and stored in a database with relevant information. Such texts are used in future for retrieval of video clips based on any given keyword. This paper shows comparative result of RGB and HSV based approaches using different types of domain and shows the result of text tracking in different scenarios.

Keywords: Tracking of text in video, Video indexing and retrieval, HSV based approach, neural network, Text recognition.

1 Introduction

Text tracking from a video is now recognized as one of the key components of video indexing and retrieval systems. Text is considered to be a strong candidate for use as a feature in high level semantic indexing and content-based retrieval as text has compact, distinctive visual characteristics i.e., a set of symbols with distinct geometrical and morphological features. An index built using extracted and recognized text enables keyword-based searches [1],[2] on a multimedia database. Text in a video lasts for a few seconds so that viewers can get adequate time to read the contents. So, any text which occurs in a video must persist in consecutive frames.

Text in image and video can be classified into two broad types: (i) stationary text (ii) moving text. Stationary text does not change their positions with time. In contrast, horizontally aligned texts with a simple linear motion may exhibit either horizontal or vertical movement across the screen in video.

Some examples of text instances and their behaviors in video are shown in Fig. 1 (a) through (b). Fig. 1(a) left shows vertical scrolling of movie credits on static background. The text lines move from bottom to top across the frames. In second the

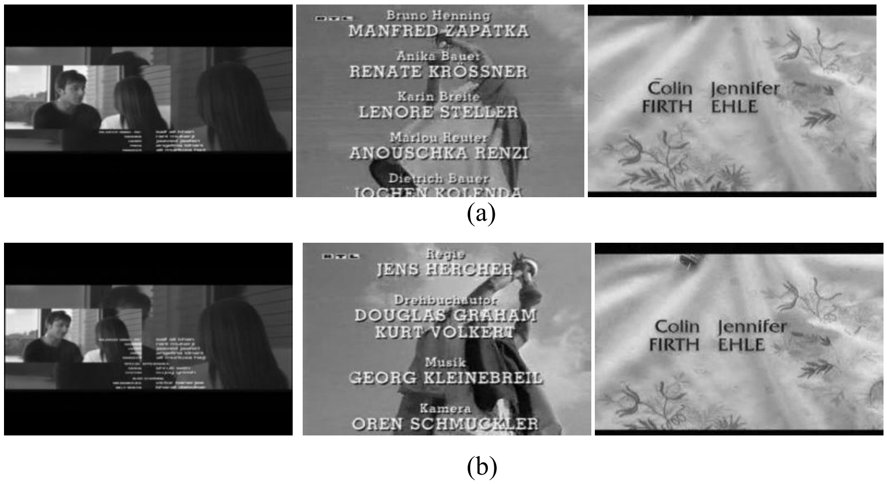


Fig. 1. Examples of video texts of different classes (a) Initial frame (b) After text or background movement

movements of text along with background are shown. Here the texts move from bottom to top direction. In third image, we show static text with slowly moving backgrounds. Our main focus is to accurately track and extract text blocks from video sequences, recognize the texts using an optical character recognizer (OCR) and store them as keywords in a database for indexing and future retrieval. The text in video sequences can be stationary or moving with simple horizontal or vertical scrolling motions across the scene. The background of the text sometimes varies frame-to-frame i.e. moving background.

2 Related Work

In recent years, a great deal of attempts have been made to develop methods for extracting text blocks from still images and videos for specific applications like license plate recognition, image and video indexing, text to speech conversion, etc.

Li et al. [3] present a novel text tracking system to detect text with simple linear motion or complex non-linear motion. They use sum of squared difference (SSD) method to track the moving text with simple linear motion and stationary text**.** For finer refinement of text tracking in complex motion, Canny edge based contour stabilization method is applied. They apply Canny operator to detect edge map of text component and finally horizontal smearing process is used to group the text block. CC based method is employed to derive the refined text position. The authors pointed out that the tracking time is 0.2 seconds per frame. Sato et al [4] investigate superimposed caption recognition in news videos. They use a spatial filter to localize the text regions as well as size and position constraints to refine the detected area. This algorithm can be applied only in a specific domain, namely, news video analysis.

Lienhart and Wernicke [5] apply forward tracking system to detect text in video. They use vertical and horizontal projection profile to generate characteristics signature from detected text line and compare with reference signature of next frame in order to find the position of text line. The authors mention that their method is sensitive to fading and cannot perform if the text persists for a long span.

Malababic et al.[6] detect artificial text in videos using a feature that captures foreground to background contrast, density of short edges of varying orientations and concentration of short vertical edges that are horizontally aligned. Various geometrical constraints are also applied for improving the result. Byun et al. [7] have presented a text extraction method using color clustering and CC analysis from scene images. They apply separate approaches for color and gray images to detect candidate text regions. For color images, geometrical clustering is used to group the same color as a preprocessing step.

The rest of the paper is organized as follows. In the next section, we give a description of our system. The results are presented in section 4 and we conclude in the last section of the paper.

3 Overview of the Approach

3.1 Shot Detection

In the context of text tracking from video, a video clip is decomposed into a number of shots. In Fig 3. we have depicted our proposed approach. Shot detection is the preprocessing step to track text in a video**.** In this section, we present an algorithm to detect shot boundary in a video using a histogram based approach. A standard way of generating a color histogram is to concatenate the higher order two bits for each of the red (R), green (G) and blue (B) values in the RGB space. A 24-bit color image is bit dropped to a 6-bit image after color reduction. Next, a two dimensional histogram is generated from the color reduced image where each row in the histogram (H) represents frequency of color pixels in the corresponding row of the video frame. Now to compare the similarities between two successive frames, histograms are generated from color reduced video frames and normalized mean centered correlation is computed. The correlation (I_m) between the frame M and next frame N is computed from histograms H_M and H_N as follows:

$$I_m = \frac{\sum_i (H_M(i) \times m_M)(H_N(i) \times m_N)}{\sqrt{\sum_i (H_M(i) \times m_M)^2} \times \sqrt{\sum_i (H_N(i) \times m_N)^2}} \quad (1)$$

H_M and H_N are two dimensional histograms of the color reduced video frames M and N, respectively. m_M and m_N represent mean values of the histograms. The correlation value less than a threshold T_s indicates shot change and a higher correlation value signifies similar frames, i.e. frames belonging to the same shot.

3.2 Key Frame Selection

Key frames are extracted from videos as representative frames to capture the salient characteristics of that shot. We use a key frame selection scheme which assumes an input video clip V as temporal sequence of images with redundancy. It has been observed that for video sequences, a text must maintain temporal stability so that viewers can read the contents. Therefore, the text is monitored in a video shot at an equally spaced video frame. In our key frame selection method, since we use only I-frames for text extraction from videos with the typical IBBPBBPBBPBB sequence at a rate of 30 frames per second, V may be represented as V= $\{I_n;\ n=1,2,.....M\}$ where I_1, I_2, ... I_M are the I-frames in the video shot and M is the total number of frames. Any text, which occurs in a video for duration less than the time gap between successive I-frames, is not useful to the viewers as well, and hence need not be considered. If a video follows any other frame sequence, we extract every twelfth frame for text extraction. It should be noted that at this stage, we target only a few key frames among large number of redundant frames. We consider the I-frames to speed up the text tracking in a video.

3.3 Frame Based Text Localizer

Color reduction is an important pre-processing step for text extraction from complex still images and videos. The mapping of RGB space to HSV space is the initial step for HSV based color reduction. A three dimensional representation of the HSV color space may be considered as a hexacone, where the central vertical axis represents intensity. Hue lies in the range [0, 2π] relative to the red axis with pure red at angle 0, pure green at 2π/3, pure blue at 4π/3 and pure red again at 2π. Saturation is the depth or purity of the color and it is measured as a distance from the central axis with the value between 1 at the outer surface for a completely saturated color and 0 at the center, which represents a completely unsaturated color. Each image can be represented as a set of triplets as follows:

$$I = \{(\mathbf{pos}, [t \mid g], Val)\} \tag{2}$$

Here **pos** denotes position of the pixel, [t | g] signifies whether the pixel is a “true color” or a “gray color” component. We consider either its hue or intensity value as the prevailing feature based on its saturation. The algorithm for computing the “true color” and “gray color” may be written as follows:

```
For each pixel in the image
    Read the RGB value
    Convert RGB to HSV
    If ((S<Min_Sat)or(V<Min_lum_fctr*no_of_color))
            V= Round (V/DIV_FCTR+57)
    Else if ((H > float (0.0))
            H= Round (H* MUL_FCTR)
    Else
            V= Round (V/DIV_FCTR+57)
```

Here no_of_color is the maximum intensity, usually 256 and Min_Sat is the minimum saturation value. Min_lum_fctr is the minimum luminance factor. Min_Sat and Min_lum_fctr are assigned values 0.2 and 0.25 respectively and work well for our experiments. In Fig. 2. original image and the corresponding HSV based color reduced images are shown. We next determine the Regions of Interest (ROIs) - Regions in the image where text could potentially be located using horizontal projection profile analysis like [8]. This step, while meant to speed-up subsequent searches, should not filter out the text regions. Care is, therefore, taken to ensure that only those regions that are certainly of type non-text are eliminated.

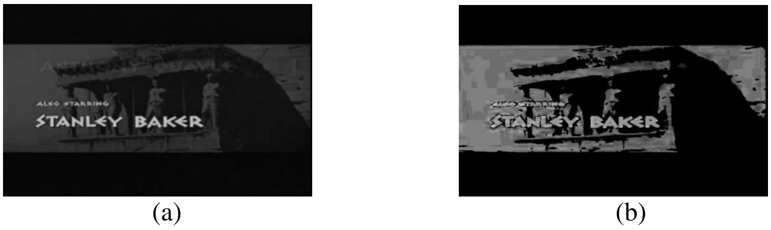


Fig. 2. Color Reduction using HSV based method (a) Original image (b) HSV based color reduced image

After identification of the regions of interest, a set of geometrical and morphological features are extracted from each ROI. We use F-ratio based method to determine the contribution of each feature in the feature set. Contribution of a feature is defined as the feature's impact on the discrimination of text and non-text classes. F-ratio can be defined as a ratio between variance of means between classes and the mean of variances within class. The criteria for better separation between text class and non-text class are increase in distribution of mean value between text class and non-text class or narrow gap in value within each class. F-ratio values of ten features which are computed from 75 text and 75 non-text data. The higher value of F-ratio indicates better impact on classification process. Next Singular Value Decomposition (SVD) method is applied on the training data set to select the optimum feature set for our feature based classifier. Singular Value Decomposition can be expressed as follows:

$$S= SVD(X) \tag{3}$$

Here X is the data set, containing features values of text and non-text training samples and S indicates the returned vector of singular values for the feature set and we consider seven features as final set for our application. A multilayer perceptron (MLP) is used to determine if the ROI contains text or non-text blocks. During the training phase, the MLP is trained with sample text and non-text blocks. During classification, the MLP uses the learnt weight values for marking each ROI. It should be noted that at this stage, we identify an entire ROI to either belong to a text region or to a non-text region and not its individual components. 200 text regions and an equal number of non-text regions are used for training the MLP. The MLP contains 7 inputs, one hidden layer of 10 units and 1 output. After classification of an ROI as text or non-text, the potential text regions are subjected to a connected component analysis for reducing the false positives.

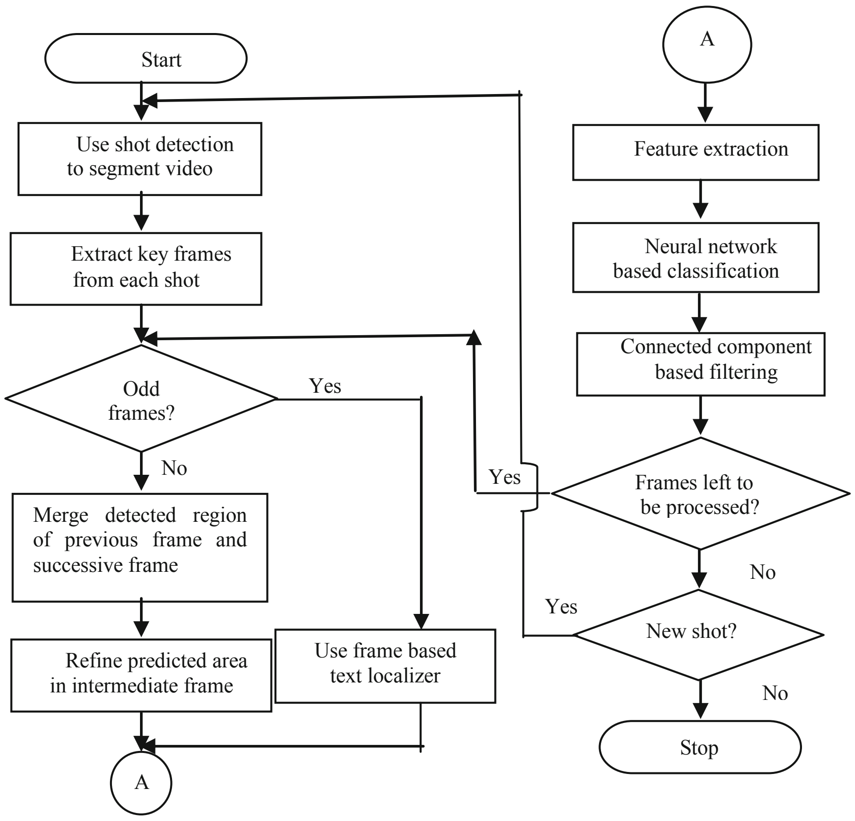

Start
Use shot detection to segment video
Extract key frames from each shot
Odd frames?
Yes
No
Merge detected region of previous frame and successive frame
Refine predicted area in intermediate frame
Use frame based text localizer
A
A
Feature extraction
Neural network based classification
Connected component based filtering
Frames left to be processed?
Yes
No
New shot?
Yes
No
Stop

Fig. 3. Flow chart of the proposed system

Connected components of the regions of interest so far marked as text, are examined for the existence of specific text features. If such features are not present in the connected components, they are eliminated. The remaining components are marked as text blocks. These text blocks are next given as input to an OCR after converted to binary image. For making it suitable to OCR input we have done some preprocessing like color polarity detection, removing connected characters and noise. The OCR output in the form of ASCII characters forming words is stored in a database as keywords with frame reference for future retrieval. In Fig. 4(a) (b) and (c) we show the original image, binarized image and OCR output.

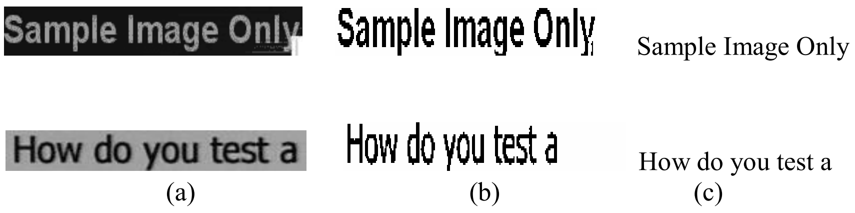


Fig. 4. (a) Image with ROIs identified (b) Binarized text block (c) OCR output

3.4 Text Localization in Key Frames Using Bi-directional Prediction

We adopt a two-stage text detection scheme to detect text in key frames. This enhances the time performance and reduces computational complexity. Above image based text localizer scheme is applied on odd key frames in a shot. We use bi-directional predictive method to detect the potential text regions in the intermediate key frames of each video shot, as there is a chance that the same text appears in the same location or with minute movements. In the even key frames or intermediate key frames, text instances are detected by forward and backward prediction. Co-ordinates of the detected text regions of previous key frame and successive key frame are merged to get the candidate text regions of the intermediate key frame. Due to motion of text instances in video, the co-ordinates may be different. The predicted regions are considered as different, if the percentage of the merged area is less than threshold value t_m. Otherwise, minimum row and column value and maximum row and column value among the predicted regions are considered as final co-ordinates of the refined predicted text region. After refinement of predicted text region in intermediate key frame, geometrical and morphological features are extracted for each predicted area. An MLP based classifier is employed to identify whether predicted text region actually belongs to text or non-text. After classification, refined text regions are subjected to a connected component analysis for reducing the false positives. Connected components of the regions of interest so far marked as text, are examined for the existence of specific text features. If such features are not present in the connected components, they are eliminated. The remaining components are marked as text blocks.

4 Performance of the Proposed System

4.1 Shot Detection Accuracy and Speed of Processing

In this section, we present quantitative performance of our text tracking system. Experiments were carried out on a dataset of 27 video clips of different categories like cartoons, TV news, educational video lectures and sports as shown in Table 1. All the video sequences are encoded in MPEG format. Our proposed system can process more than 18 frames per second. Our system can process a frame in 0.11 second using Intel Core i3 2.26 GHz machine. In comparison [9] take 1.2 sec for processing a frame of same size using Sun Ultra Sparc 60 and [10] take 0.47 for processing a frame of size 320 X 240 and Our proposed method is very fast since most of the computationally intensive algorithms are applied only on the regions of interests. Table 1 shows the accuracy of hard cut detection on a variety of video sequences. More than 96% accuracy is achieved for commercials, news and video lectures. For soccer videos the shot detection accuracy is less than 88%. Due to movements of players although the scene changes in the video, there is only little variation in the color distribution of the histogram.

4.2 Text Detection Accuracy

Video clips of 35 minutes are extracted from TV news, movie credits, video lectures and other sources. Operational time of the proposed system and experimental results are shown with video clips of different categories. We use MPEG video sequences at a resolution that varied between 352X240 and 384X288 with RGB values of 256 levels. Frame rates of videos are in the range of 25 to 30 frames per second. We successfully detect text of the classes as enumerated below.

(i) Static text with static background
(ii) Static text with moving background
(iii) Horizontal scrolling text with static background
(iv) Vertical scrolling text with static background
(v) Horizontal scrolling text with moving background
(vi) Vertical scrolling text with moving background
(vii) Text with special effects

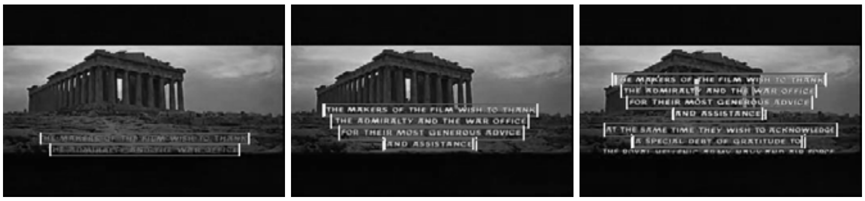
Table 1. Accuracy of shot detection in MPEG video sequences

Type of Video	No. of Video Clips	No. of Frames	%Of Shots Detected Correctly
News	5	17000	96.8%
Movies	8	13300	98.3%
Sports	2	12000	82.3%
Video Lectures	7	16600	96.8%
Advertisement	5	11700	87.6%

Another important consideration is the quality and complexity of pictures for evaluation. In [11] large fonts in web images, advertisements and video clips are considered. Kim [12] does not detect low contrast text and small fonts and [3] use text with different complex motions. Zhang et al [10] as well as [4] detect only caption text in news video clips. We are able to detect text under a large number of different conditions as enumerated below.

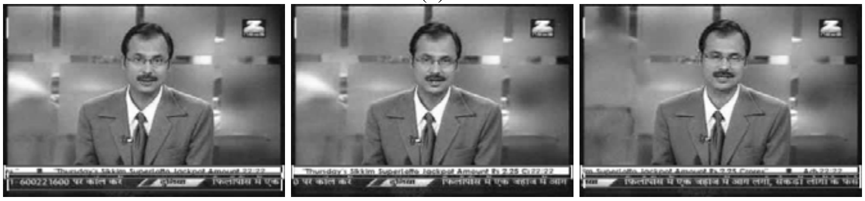
Fig. 5(a) shows the performance of text tracking in a video sequence with vertical scrolling movement of texts on a static background. Here the text lines consist of small font and they are shifting from bottom to top across the scene. Fig. 5.(b) illustrates tracking performance in a news video. The news headlines move fast from right to left across the scene. The news captions are in English and Hindi languages and both fonts are detected correctly. In Fig. 5(c) we demonstrate tracking of subtitles from a movie clip. In this case the background is changing and the texts are composed of small fonts. The text instances are tracked efficiently by the algorithm though there are movements in background.

In Fig. 6. we show performance of tracking algorithm for a movie clip with high noise. The text appears randomly on a static background and there are presence of fading in and fading out effects of text instances. In frame numbers 11, 22 and 33 the text instances are fading in. Fading out of text instances are shown in frame number 45, 56 and 63.


THE MAKERS OF THE FILM WISH TO THANK
THE ADMIRALTY AND THE WAR OFFICE
FOR THEIR MOST GENEROUS ADVICE
AND ASSISTANCE
AT THE SAME TIME THEY WISH TO ACKNOWLEDGE
A SPECIAL DEBT OF GRATITUDE TO

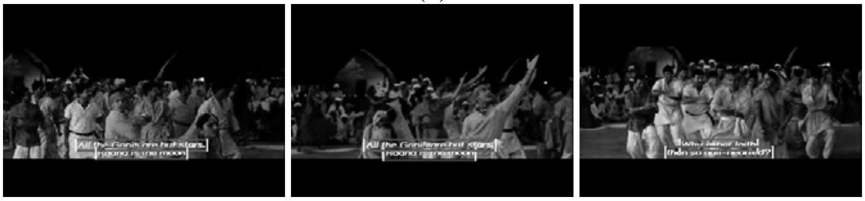
Frame 15 Frame 105 Frame 195

(a)



Frame 3 Frame 30 Frame 63

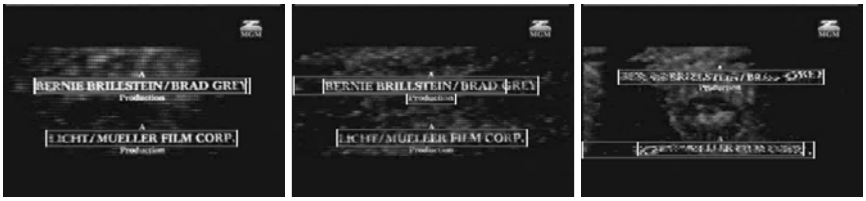
(b)



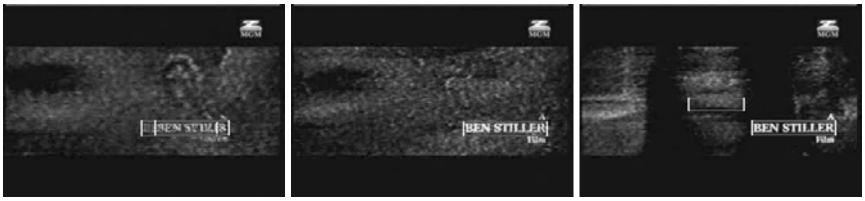
Frame 22 Frame 58 Frame 93

(c)

Fig. 5. (a) Tracking algorithm applied to caption text with vertical scrolling (b) Detection of horizontal scrolling news caption (c) Results of text tracking with dynamic background



Frame 11 Frame 23 Frame 35



Frame 45 Frame 56 Frame 67

Fig. 6. Detection of text events with high noise and fading effect

Table 2. Accuracy of text tracking for different combinations of text and background movements

Type of Text Instances	Video Duration (hh:mm:ss)	No. of Text Lines	%of Text Detected Correctly
Static text with static background	00.04:20	12	98%
Static text with moving background	00:07:05	16	87%
Horizontal scrolling text with static background	00:14:10	30	97.3%
Vertical scrolling text with static background	00:05:40	38	92.4%
Horizontal scrolling text with moving background	00:02:30	15	91.4%
Vertical scrolling text with moving background	00:02:10	16	82%
Text with special effect	00:02:08	9	77.6%

In Table 2, we show the accuracy of text tracking with a number of video clips of different combinations of text and background motions. It is seen that detection performance is better for static background than moving background and more than 92% detection is achieved in case of static background. These video sequences total about 35 minutes with a wide variety of font, color, domain, motion and complexity. The video sequences consist of 463 words, 127 lines in total.

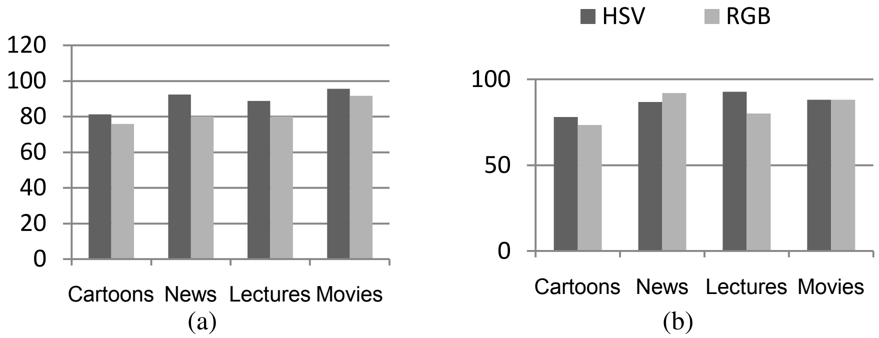


Fig. 7. Performance comparison of HSV and RGB based approaches with different types of video clips (a) Recall (b) Precision

The performance shows that text tracking approach works well in both stationary text as well as moving text with simple linear motion. However, our proposed methodology cannot detect text with very complex motion and skewed text.

In every text rich image or video-frame, there are two regions, text dominated and non-text dominated. It is also very important not to detect any non-text part as text part. The performance can be measured in terms of recall and precision. They can be defined as follows Recall can be defined as the ratio of text regions identified correctly as text regions and total no of text instances in the frame and similarly Precision is defined as ratio of correctly identified text regions and total no of detected text region by our proposed system.

We have calculated recall and precision on large and diverse sets of text-rich video frames for HSV based approaches. For video frame processing, we have tested the system on different types of domains such as news clips, lecture clips and movies and cartoons. We have made comparison with [8] and in Fig.7 (a) and (b) we show comparative recall and precision values of HSV and RGB for different sets of test data of different domains. To the best of our knowledge, this is the first successful attempt for text detection taking different types video with moving text. It should be noted that our methods are robust and stable for unconstrained videos and the system parameters remain the same throughout all the experiments.

4.3 Processing Speed

We next study the effect of shot detection on the operational time of proposed text tracking system. It may be argued that the time spent in shot detection is more than the ROI processing time saved. In Table 3, we compare the running time of two versions of the algorithm, one with shot detection, and one without. It can be seen that more than 45% time is saved by using shot detection. The time performance given in Table 3 has been achieved on a Intel Core i3 2.26 GHz machine.

Table 3. Effect of shot detection and text tracking on running time

Video Duration (hh:mm:ss)	**Running Time(hh:mm:ss)**			
	Without Shot Detection	**With Shot Detection**		
		Shot Detection+ Key Frame Selection	**Text Tracking**	**Total Time**
00:00:30	00:06:35	00:02:55	00:00:20	00:03:15

5 Conclusion

We have built a complete automated content based video retrieval system using embedded text. We integrate all the modules starting from the detection of the text until the keywords are converted to ASCII characters stored in a database with a link to the video sequence and the frame number for future retrieval. A great deal of work will have to be done to make the system more efficient. First, we plan to extend our work in the compressed domain processing to make it even faster. Secondly, our system cannot detect non-horizontal text in an image. Thirdly, we have to investigate possible improvements of our method for better tracking of text with complex motion. Fourthly, the recognition accuracy of our system is poor for text with complex background. Finally, a more accurate OCR will also improve the quality of retrieval further.

References

1. Niblack, W.: The QBIC Project: Querying Images by Content Using Color, Texture and Shape. In: Proc. Storage and Retrieval for Image and Video Databases, pp. 173–187. SPIE, Bellingham (1993)
2. Ma, W.Y., Manjunath, B.S.: Ne Tra: A Toolbox for Navigating Large Image Databases. Journal Multimedia System, 184–198 (1999)
3. Li, H., Doerman, D., Kia, O.: Automatic Text Detection and Tracking in Digital Video. IEEE Transactions on Image Processing 9, 147–156 (2000)
4. Sato, T., Kanade, T., Hughes, E., Smith, M.: Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Captions. Multimedia Systems 7, 385–394 (1999)
5. Lienhart, R., Wernicke, A.: Localizing and Segmenting Text in Images and Videos. IEEE Transactions on Circuits and Systems for Video Technology 12, 256–268 (2002)
6. Malobabic, J., O'Connor, N., Murphy, N., Marlow, S.: Automatic Detection and Extraction of Artificial Text in Video. In: Adaptive Information Cluster, Center for Digital Video Processing, Dublin City University (2002)
7. Byun, H.-R., Roh, M.-C., Kim, K.-C., Choi, Y.-W., Lee, S.-W.: Scene text extraction in complex images. In: Lopresti, D.P., Hu, J., Kashi, R.S. (eds.) DAS 2002. LNCS, vol. 2423, pp. 329–340. Springer, Heidelberg (2002)
8. Misra, C., Sural, S.: Content Based Image and Video Retrieval Using Embedded Text. In: Narayanan, P.J., Nayar, S.K., Shum, H.-Y. (eds.) ACCV 2006. LNCS, vol. 3852, pp. 111–120. Springer, Heidelberg (2006)
9. Wong, E.K., Chen, M.: A New Robust Algorithm for Video Extraction. Pattern Recognition 36(6), 1397–1406 (2003)
10. Jung, K., Han, J.H.: Hybrid Approach to Efficient Text Extraction in Complex Color Images. Pattern Recognition Letters 25, 679–699 (2004)
11. Jain, A.K., Yu, B.: Automatic Text Location in Images and Video Frames. Pattern Recognition Society 31, 2055–2076 (1998)
12. Kim, H.-K.: Efficient Automatic Text Location Method and Content-Based Indexing and Structuring of Video Database. Journal of Visual Communication and Image Representation 7, 336–344 (1998)

Enhanced Insider Threat Detection Model that Increases Data Availability

Qussai Yaseen and Brajendra Panda

Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, AR 72701, USA
{qyaseen,bpanda}@uark.edu

Abstract. This paper demonstrates how to prevent or mitigate insider threats in relational databases. It shows how different order of accesses to the same data items may pose different levels of threat. Moreover, it states the conditions that are required to regard a data item as expired. In addition, it introduces the two different methods of executing insiders' tasks, and how to prevent insider threat in those. The models presented in this paper organize accesses to data items in a particular sequence so that the availability of data items is maximized and the expected threat is minimized to the lowest level. Furthermore, it determines when to give an insider an incorrect but acceptable value of a risky data item in order to prevent a possible threat.

Keywords: Insider Threat, Databases, Knowledgebase, Data Availability.

1 Introduction

In the information age where both information systems and attacks on them have become more complicated, preserving the security of sensitive data has become a major research issue. While most of research has focused on defending systems against hackers or outside threat, threats coming from insiders are becoming more risky and damaging. Surprisingly, insiders were responsible for 52% of all security breaches in 2004 [1]. Such threats are harder to detect due to the fact that insiders exploit their privileges and familiarity with the properties and internal structure of the system to launch attacks. In a relational database system, which is the focus of this work, insiders are usually familiar with the schema, dependencies, and other properties of the database. This familiarity gives them an advantage over outsiders and makes an attack lunched by such an insider difficult to detect.

To date, extensive research has been performed on detection of attacks originating from outside the system while very little has been done in the area of insider attack detection. Moreover, mechanisms developed for the former are not effective for the latter since removing a suspect in the latter case would result in removal of a valid user of the system, and this in turn would affect the performance of the organization. While most of the research performed in the area of insider threat has been done at the system level, very little progress has been made at the application level such as the database systems.

This paper, which is an enhanced version of our previous work [2], discusses the insider threat issues in relational databases, and proposes mechanisms to mitigate or prevent the problem. We state the conditions required to regard the lifetime of a data item as expired and introduce two methods for executing an insider's task, namely, batch of transactions and transaction by transaction methods. Furthermore, we explain techniques to prevent or mitigate the insider threat in both cases. Our models use the Neural Dependency and Inference Graph NDIG to make the prevention of insider threat more effective in situations when an insider has to get access to risky data items.

The rest of the paper is organized as follows. The next section discusses some related work. Section 3 explains the threat due to the insiders' knowledgebase as well as the lifetime of data items. Section 4 describes the effect of organizing insiders' accesses to data items. Section 5 presents the methods for computing the risk of possible sequences of data item accesses and how to choose the lowest risk sequence. Finally, section 6 offers the conclusions and the plan for future work.

2 Related Work

Several definitions of the term "insider" have been introduced in [3][4][5]. Yaseen and Panda [5] defined the term "insider" in the context of relational databases as "the person who has privileges to access and is familiar with the structure of the system under consideration, and is inspired to harm that system". This definition summarizes the problem and the power of insiders.

Some researchers have used existing methods of detecting external threat, such as using honeypots [6], to recognize insider threat while others have introduced new techniques to detect and prevent such attacks, for example, Althebyan and Panda [7]. The latter introduced the knowledge graph of an insider at the system level as well as the dependency graph for the data item accesses, and used them to detect and prevent insider threats. However, their work was at the system level and did not consider relational databases.

Insider threat in relational databases is strongly related to dependencies among data items since insiders can use this knowledge to get unauthorized information and make malicious changes. The research by Farkas et al. [8][9] discussed the problem of combining non-sensitive data to get sensitive data using dependencies. Farkas et. al [9] showed how updates can affect the knowledgebase of users. In addition, they demonstrated that looking through the history of accesses of users without checking the lifetimes of the data items limits the availability of data items.

Researchers in [10] and [11] discussed the inference channel. Yaseen and Panda [5] introduced new graph called the Neural Dependency and Inference Graph (NDIG) that shows dependencies and the amount of information that can be inferred about data items using dependencies. In [14] the authors showed how insiders can use dependencies to make unauthorized changes to data items.

3 The Effect of Knowledgebase and Lifetime of Data Items

The knowledgebase of an insider contains the values of data items that he/she has accessed. These values may be combined with some insensitive data items that he/she may request later to infer sensitive information, which pose a threat [5][12]. Revoking read accesses from previously accessed data items does not eliminate the threat since the values still exist in the insider's knowledgebase. For instance, consider the dependency [Rank, Experience] $\rightarrow$ [Salary]. If an insider had accessed the Rank attribute and he/she is given a read access to the Experience attribute, he/she can infer the value of the Salary attribute, which could be sensitive information.

Clearly, an insider's knowledgebase could pose a serious threat, but not if the data items in the knowledgebase are expired. That is, if other insiders modify the data items whose values may also exist in the insider's knowledgebase, the lifetime of those data items (old values) would expire. Thus, using them to infer sensitive information would not pose a threat. In light of this, determination of the lifetimes of data items in an insider's knowledgebase is important.

However, merely updating values of data items does not always make their lifetimes expire. To clarify this point, we should mention the types of dependencies explained in [12]. The method that is presented there classified dependencies based on different categories. One of these categories is the strength of the dependencies, which is classified into two types: strong and weak. A dependency between two data items A and B is called a strong dependency if a change in A forces a change in B. For instance, the dependency [Rank $\rightarrow$ Salary] is a strong dependency since any change in the Rank attribute makes a change in the Salary attribute. On the other hand, a dependency between two data items A and B is called a weak dependency if a change in A may or may not make a change in B. For instance, the dependency [Score $\rightarrow$ Grade] in a student table is a weak dependency since a change in the Score attribute does not always make a change in the Grade attribute. To clarify this point, suppose that a student gets A in a class if he/she gets a score between 90 and 100, and gets B if he/she gets a score between 80 and 89. Now, if the score of a student is updated from 85 to 87, the value of his/her grade does not change. However, if the score of the student is updated from 88 to 91, his/her grade changes from B to A. The inference is based on the familiarity of insiders of the dependencies and their constraints that exist in the database [12]. That is, in case of the example above, if an insider gets access to the Score attribute of a student and the insider is familiar with the dependency constraints, he/she can infer the Grade of a student without having access to it. For instance, suppose that the insider has read a student's score, say 85, he/she infers that the student's grade is B. However, suppose that the student's score has been updated to be 88 and the insider is prevented from accessing the student's score again. In this case, the insider still infers the right value of the student's grade based on the old value of the student's score. We say in this case that the old value of the student's score in the insider knowledgebase has not been expired although it has been updated. In light of this discussion, we define the lifetime of data items as follows.

Definition 1: Given the data items A and B in a relational database DB and the dependency A $\rightarrow$ B, the *lifetime* of the data item A expires when it is updated to a value such that if an insider uses the old value of A to infer information about B, his/her inference will be incorrect.

To understand the role of data life-time, let us consider the following. Suppose the security protocol denied the request of an insider to access a data item, say K, due to the reason that the insider may combine it with a data item, say R, that is in his/her knowledgebase to infer unauthorized information. However, if the value of data item R has expired, the system unnecessarily denied the access to K since providing the value of K would not create a problem; rather by denying access to K, delays the user from performing his/her job on a timely basis. Similarly, ignoring the knowledgebase and granting access irrespective of the history of previous accesses may pose a threat. Thus, both these issues should be considered when a user requests accesses to data items. The work by Farkas et. al [9] attempts to increase the availability of data items by checking the updates history. In their work, each insider has a history file that stores all data items, which the insider either has previously received or can disclose from the received data items. When an insider launches a query, all data items that can be received from this query are stored in the file. The data items that an insider can infer are discovered by considering the current request, the history file, and the dependencies among data items. Based on the inferred data items, the system decides whether to grant or deny the requested data items. However, some data items that were accessed in the past may have been updated by others as explained before. Therefore, the inferred data items based on those expired data items would be incorrect.

4 Ordering Access Sequence

A task of an insider may consist of several operations involving many data items. Some operations may have dependent relationships, i.e., they should be performed in some order, while other operations are independent and can be performed without any order among themselves. In some situations, different order of accesses imposes different levels of risks. This section demonstrates how to prevent insider threat and increase the availability by choosing a safe sequence of operations.

An insider may perform his/her tasks in one of the two ways: he/she may submit his/her task as a group of transactions representing the entire task or submit the transactions one after another.

4.1 Tasks as a Batch of Transactions

This approach considers various insiders in the system, their tasks, the set of operations required for each task, and the dependencies among the operations. It must be noted that investigating the knowledgebase of each insider is a major undertaking. Our approach helps in organizing insiders' accesses to data items in order to reduce the risk level and to mitigate the effect of past requests on current transaction.

Figure 1 shows a system at a given point of time with current set of insiders S_1 to S_6. For insider S_1, it shows the sequence of accesses that S_1 requests to perform his/her task. The data item L is in the knowledgebase of S_1. Dashed arrows represent the situation that if S_1 gets access to the data item C, he/she can combine it with L to get sensitive information about the data item T to which he/she has no access privilege; this indicates a threat. As shown in the figure, insider S_1 needs to get access to data items Q and V to work on the data item R. Similarly, he needs access to C to work on K and so on. Notice that the insider may be given access to C before allowing

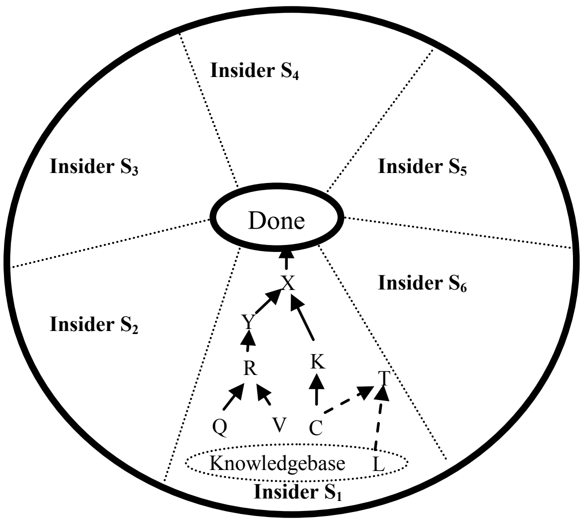


Fig. 1. A Snapshot of a System Showing Current Insiders and Tasks and Knowledgebase of one of them

access to Q or V as they do not have any dependency relationship among them. However, he must work on R before working on Y since the latter is dependent on the former. Determining which accesses the insider should get first depends on the level of the risk that the access sequences pose and how the decision affects availability of data items. Granting accesses randomly may increase the risk and reduce the availability as explained earlier.

To minimize the risk that insider S_1 may pose to the lowest level, the insider should not get read access to C until the lifetime of L expires. This prevents the insider from inferring correct information about T. However, delaying the insider's task until the lifetime of L expires is not always a good solution since the delay may continue for long. To solve this problem, the insider is given access to other data items to work on an independent operation until the lifetime of L expires. In this example, the insider may get access to Q and V to work on R first, before he gets access to C. We can make the lifetime of L expire by giving a different insider who wants to modify L, say S_3, a write access on it. Thus, the lifetime of L would expire after it is updated by S_3. In this case, after S_1 finishes his/her work on R and S_3 updates L and makes it expires in the knowledgebase of S_1, giving S_1 access to C may pose a much lower threat or would not pose a threat at all. Obviously, this increases the availability of data items. Moreover, it mitigates the threat due to the values present in the knowledgebase of insiders.

The following example explains how the proposed approach works. Consider the three relations in a relational database: Table 1, Table 2, and Table 3. Suppose that the database has the functional dependency {Rank $\rightarrow$ Base_Salary}. In addition, sssume that the data items (Name, Rank), (Rank, Base_Salary), and (Name, Experience_in_the_Rank) are not sensitive information, while the data items (Name, Base_Salary) and (Name, Salary) are sensitive information. In addition, suppose that the salary of an academic staff is computed using the formula: Salary = Base_Salary + 200 * Experience.

Table 1. Professor Table

ID	FName	LName	Rank	Experience_in_the_ Rank	DeptID
3301	George	Thompson	Assistant Prof	3	154
3302	Jamal	Yaseen	Full Prof	2	452
3303	Jiff	Tyson	Associate Prof	5	154

Table 2. Rank_Salary Table

Rank	Base_Salary
Assistant Prof	100K
Associate Prof	120K
Full Prof	140K

Table 3. Department Table

DeptID	Name	Location
168	Computer Science	SSED
597	Electrical Engineering	LKEF

Now, assume that there are two insiders who are currently working in parallel, where the task of the first insider (Insider1) is as follows:

Query 1: Retrieve the name and the rank for all computer science professors.
Query 2: Retrieve the experience in the rank for the professor Jiff.
Query 3: Retrieve the Base_Salary of all associate professors.

While the second insider (Insider2) has the following task.

Query 4: Promote Jiff to Full Professor position.

Suppose that Insider1 launches his/her first and second queries (Query1 and Query2). Obviously, executing these two queries does not pose any threat. Thus, the system allows the insider to get this information, which is: (< James, Assistant Prof >, < Jiff, Associate Prof >, < Jiff, 5>).

Now suppose that Insider1 launches the third query (Query3). If Insider1 gets the privilege to execute this query, he/she will get the information (< Associate Prof, 120K>). In this case, the insider can combine this information with the data items he/she retrieved from Query1 and Query2 to get the sensitive information <Jiff, 120K>, which is a threat.

If the system discovers this threat and declines Insider1's request, Insider1's task will be affected negatively. On the other hand, if the system does not discover this possible threat, sensitive information may be revealed. Thus, both cases affect the system negatively.

Next, let us consider a scenario for satisfying the requests of the two insiders. Suppose that the query Query4 is executed before the query Query3. This means that Insider2 promotes the rank of "Jiff" from associate professor to full professor before Insider1 gets access to the Base_Salary of associate professors. In that case, after Insider2 executes Query4, Insider1 would infer incorrect information about Jiff's sallary. This means that the sequence <Query1, Query2, Query3, Query4> pose a threat, while the sequence <Query1, Query2, Query4, Query3> does not pose any threat. This example demonstrates the importance of choosing the order of executing the requested operations in preventing the insider threat and increasing the availability of data items.

4.2 Limitations and Possible Solutions

Organizing access sequences as discussed earlier either eliminates or significantly reduces the risk of some data items present in the knowledgebase of an insider. This is performed by letting other insiders modify the data items so that their previous values are expired before they are read by some other insiders. However, what can be done if there is no other user who requests a modification of such a data item? To solve this problem, the granting of risky accesses may be delayed until such modifications are performed. But, this method would result in data unavailability for users and degraded system performance. Moreover, if the insider must get access to the requested data item to perform his/her job on a timely manner, the above mentioned solution is unacceptable. In this case, we may grant the insider an incorrect value of the risky data item and correct the results later based on the correct value of that data item. The next section discusses this idea.

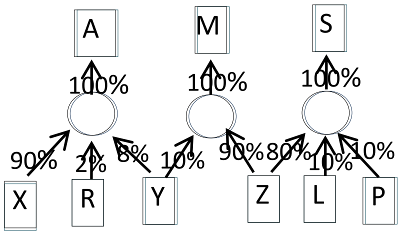


Fig. 2. A Part of an NDIG of a Database

4.3 Providing Incorrect Values of Data Items

When incorrect values of data items are provided to insiders, they will not be able to infer correct values of dependent data items. We propose to do so when the inferable data is sensitive. However, this approach may negatively affect insiders' trust about the system. To mitigate this issue, incorrect but close enough values must be provided while making sure that the values still do not disclose any sensitive data. To know how much information one can infer, we make use of the Neural Dependency and Inference Graph (NDIG) [5]. An example of NDIG is shown in Figure 2. Cyclic inference is omitted in this graph for simplicity. Suppose that an insider K had accessed the data items L and P in this database. Later, he/she requested the data item Z. Figure 3 shows K's task. Assume that the insider K's threshold is 100% for all data items except for the sensitive data item S, which is 65%. In addition, assume that the value of S ranges between 0 and 100, and it is computed using the formula: $S = 4*Z + L + P$.

Obviously, using the proposed approach, the insider is given first an access to the data items X, R, and Y to work on M. He/she is not given access to Z first because he/she can combine it with the data items L and P, which are in his knowledgebase, to get sensitive information more than the allowable threshold about the data item S. Suppose that at the time, there is no insider requesting a write access on either L or P.

In addition, assume that due to the time sensitive nature of insider K's task, the system has to grant him/her the access to data item Z. Clearly, granting the access poses a threat. Thus, to avoid this threat, the insider is given an incorrect value of Z. Notice that the given incorrect value does not mean that the value of Z is changed in the database. It means that the system provides an incorrect value to the insider. However, this incorrect value should satisfy two conditions, which are:

- It should not be very different from the correct value, otherwise this would affect the insider's trust on the system, if the insider has a guess on the range of the value.
- Using the value, the user should not be able to make a correct estimation of the sensitive data item.

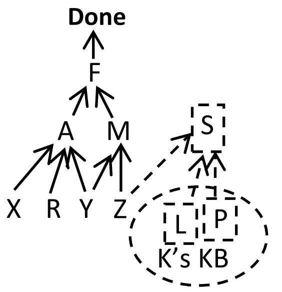


Fig. 3. Insider K's Task and Knowledgebase

After giving the insider an incorrect value of Z, the system should track the subsequent modifications on the data items that the insider K makes using the incorrect value of Z and correct those using the right value of Z. This process applies to other users who access such damaged data items as well. Damage assessment and recovery are not the focus of this paper and, therefore, the methods will not be discussed here. As a reference, interested readers may review the work presented in [15].

4.4 Tasks Executing a Transaction at a Time

There may be situations when the system does not know what task an insider plans to perform. This is the case when the insider submits one transaction after another rather than submitting all the transactions to complete his/her task. Hence, the system cannot use the previous approach. To solve this problem, accesses patterns for each task of each insider can be extracted. Insiders having a specific role access the same data items to perform a specific task. However, the order of access of data items in each task may differ. Thus, for each different task, all patterns of accesses of data items should be extracted and stored. These patterns are used to construct the Task Graph, which shows the data items, the paths of possible accesses to those data items and dependent and independent operations in a task. For instance, the sub-graph with solid lines in Figure 3 is an example of a task graph.

Tasks graphs are used to predict the tasks of insiders based on what they request when they execute their tasks. Predicting the task of an insider and combining it with his/her knowledgebase, as shown in Figure 3, facilitates the prediction. The feasibility of this approach depends on the ability of the system to extract the correct patterns of the tasks at hand. In addition, it also depends on construction the correct task graph, which may not be an easy job. We, as our future research direction, intend to develop appropriate methods for this.

5 Choosing the Sequence with the Lowest Risk

After considering active insiders in the system and their tasks, the system organizes their accesses to data items to prevent any insider threat. There are many possible sequences of data accesses to complete a task. Finding a safe sequence is critical to preventing a threat. A Safe sequence is defined as follows.

Definition 2: A safe sequence is a sequence of operations when executed in that order does not reveal any sensitive information either with or without the data in the corresponding insiders' knowledgebase.

However, choosing a safe sequence of operations is not always achievable. Thus, an acceptable sequence should be chosen, which is defined as follows.

Definition 3: An acceptable sequence is a sequence that reveals insignificant sensitive information to the insider under consideration so that, even with the data in the insider's knowledgebase, it doesn't pose any intolerable threat to the system.

Security administrators decide whether the revealed information is insignificant or not, or whether it poses an intolerable threat. In order to choose either a safe or an acceptable sequence, the risk of granting each request is computed. The risk in such a case is the maximum difference between a sensitive data item that may be revealed by granting this request and the threshold value for the requested data item with respect to the insider. The following formula summarizes this, where R_j is a request by the given insider I and d_i is a sensitive data item that may be revealed by granting the request.

$$\text{Risk}(R_j)=\text{Max}(\text{Sensitivity}(d_i) - \text{Threshold}(I,d_i)). \quad (1)$$

Next, the risk of the entire sequence of requests for an insider is computed as the summation of the risks of all requests the sequence contains. During the computation the previous requests in the same sequence are also considered to see if any of the data item's values have expired, thus, posing a reduced risk. It must be observed that, therefore, the risk of a request when calculated by checking data items accessed in previous requests may differ from that when computed otherwise.

Using this method, an acceptable sequence, which poses the lowest risk among different sequences, is chosen. However, if no acceptable sequence is found, the insider's request is either denied or delayed till another user's update reduces the risk to an acceptable level.

6 Conclusions and Future Work

Insider threats cannot be accurately detected just by considering the impending request of an insider. Rather, all past data accesses must be taken into consideration during the process. Therefore, each insider's knowledgebase plays a critical role in threat computation. However, some of the data items in a knowledgebase may not have any relevance if they have been modified by another user. If the validity of such data items is not taken into account, the threat prediction model may signal a threat whereas in reality there may not be one. In order to reduce such false alarms during threat analysis, we have advocated checking the data dependency relationships and validity of data values. We have presented two approaches for safely executing insiders' tasks. Since the feasibility of our models depends on the capability of the system to accurately identify insiders' tasks, as our future research, we plan to develop a task detection mechanism and integrate that with our current models. Furthermore, to assess their effectiveness, we wish to carry out performance study of the models.

Acknowledgement. This work has been supported in part by US AFOSR under grant FA 9550-08-1-0255. We are thankful to Dr. Robert. L. Herklotz for his support, which made this work possible.

References

1. Gordon, L., Loeb, M., Lucyshyn, W., Richardson, R.: Computer Crime and Security Survey, `http://www.cpppe.umd.edu/Bookstore/Documents/2005CSISurvey.pdf`
2. Yaseen, Q., Panda, B.: Organizing Access Privileges: Maximizing the Availability and Mitigating the Threat of Insiders' Knowledgebase. In: 4th International Conference on Network and System Security, Melbourne, Australia (2010)
3. Bishop, M., Gates, C.: Defining the Insider Threat. In: 4th Annual Workshop on Cyber Security and Information Intelligence Research. Oak Ridge, Tennessee (2008)
4. Brackney, R., Anderson, R.: Understanding the insider threat. Technical Report, RAND Corporation (2004)
5. Yaseen, Q., Panda, B.: Knowledge Acquisition and Insider Threat Prediction in Relational Database Systems. In: International Workshop on Software Security Processes, Vancouver, Canada, pp. 450–455 (2009)
6. Spitzner, L.: Honeypots: Catching the Insider Threat. In: 19th Annual Computer Security Applications Conference, Washington, DC (2003)
7. Althebyan, Q., Panda, B.: A knowledge-base model for insider threat prediction. In: IEEE Workshop on Information Assurance and Security, West Point, NY, pp. 239–246 (2007)
8. Farkas, C., Jajodia, S.: The Inference Problem: A Survey. ACM SIGKDD Explorations, pp. 6–11 (2002)
9. Farkas, C., Toland, T., Eastman, C.: The Inference Problem and Updates in Relational Databases. In: 15th IFIP WG11.3 Working Conference on Database and Application Security, Ontario, Canada, pp. 181–194 (2001)
10. Brodsky, A., Farkas, C., Jajodia, S.: Secure Databases: Constraints, Inference Channels and Monitoring Disclosures. IEEE Trans. on Knowledge and Data Engineering, 900–919 (2000)

11. Yip, R., Levitt, K.: Data Level Inference Detection in Database Systems. In: 11th Computer Security Foundations Workshop, Rockport, MA, pp. 179–189 (1998)
12. Yaseen, Q., Panda, B.: Predicting and preventing insider threat in relational database systems. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 368–383. Springer, Heidelberg (2010)
13. Morgenstern, M.: Security and Inference in Multilevel Database and Knowledge-Base Systems. ACM SIGMOD Record, 357–373 (1987)
14. Yaseen, Q., Panda, B.: Malicious Modification attacks by Insiders in Relational Databases: Prediction and Prevention. In: 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust, Minneapolis, Minnesota (2010)
15. Yalamanchili, R., Panda, B.: Transaction Fusion: A Model for Data Recovery from Information Attacks. Journal of Intelligent Information Systems 23(3), 225–245 (2004)

Checking Anonymity Levels for Anonymized Data

V. Valli Kumari, N. Sandeep Varma, A. Sri Krishna,
K.V. Ramana, and K.V.S.V.N. Raju

Dept of Computer Science & Systems Engineering, Andhra University,
Visakhapatnam, Andhra Pradesh, India, 5300 03
{vallikumari,snvarma9,srikrishna.au,kvramana.777,
kvsvn.raju}@gmail.com

Abstract. Privacy Preserving Publication has become one of the most prominent research topics in the recent years. Several techniques like k-anonymity, *l*-diversity and (α, k) anonymity were proposed to preserve privacy. Most of the published work focuses on anonymizing the microdata for preserving privacy and now the focus towards the verification of the anonymity levels of the microdata before publishing is the need of the day. Many publishers claim having anonymized the data. Verification of the claim on a large anonymized dataset is a herculean task. This paper focuses on providing simple approach for checking the anonymity levels for an anonymized dataset using frequent itemset generation. A GUI based tool named PRUDENT was developed to demonstrate the practicality of the solution. PRUDENT deals with numerical, categorical and multiple sensitive attributes. Results show that the algorithm is feasible and practical. A comparison with the existing methods is shown.

Keywords: Privacy, PRUDENT, Frequent Itemsets, Vertical Data Format.

1 Introduction

With the increase in the identity theft, privacy preserving data publication took its prominent role in the research arena. Published data from organizations like hospitals, financial institutions is available for survey or mining purposes. On the other hand the individuals are unaware of the future privacy threats as their information is being shared with various vendors. This paper provides a practically feasible verification tool for validating the anonymity levels.

In 2002 Sweeney [1] discovered that the medical data was exposed when the dataset was cross referenced with the voter's registrations data. When de-anonymization was applied on the Netflix database, it exposed the individual's information [2]. In 2006 AOL [16] removed their published query logs quickly because of re-identification of the individuals from their query logs. An extensive research was done in the past decade on privacy breaches in public data. Though statistical databases were anonymized using perturbation techniques [4][6], the usefulness of the published data was limited. Personalized privacy [7] is another approach to let each individual set his individual privacy level for his data.

The main objective of the privacy preserving publication is to find a distorted table ΔT from the original microdata table T, by applying anonymization principles like k-anonymity [1], *l*-diversity [10], (α,k)-anonymity. The objective is to preserve privacy and retain as much utility as possible in ΔT.

Table 1. Original Table

Job	**Sex**	**Age**	**Disease**
Lawyer	Male	28	Cancer
Engineer	Male	25	HIV
Writer	Female	33	Asthma
Dancer	Female	32	Hepatitis
Dancer	Female	35	Hepatitis
Writer	Female	34	HIV

Table 2. 2-anonymous Table

Job	**Sex**	**Age**	**Disease**
Professional	Person	[20-30]	Cancer
Professional	Person	[20-30]	HIV
Writer	Female	[30-35]	Asthma
Writer	Female	[30-35]	HIV
Dancer	Person	[30-35]	Hepatitis
Dancer	Person	[30-35]	Hepatitis

Table 3. 2-Anonymous with *l*=2

Job	***Sex***	***Age***	**Disease**
Professional	Person	[20-30]	Cancer
Professional	Person	[20-30]	HIV
Artist	Person	[30-35]	Asthma
Artist	Person	[30-35]	HIV
Artist	Person	[30-35]	Hepatitis
Artist	Person	[30-35]	Hepatitis

Table 4. Anonymized Dataset

Id	***Age***	***Gender***	***Zipcode***	***Salary***
1	1-100	Male	510001-520000	11001-21000
2	1-100	Male	510001-520000	1001-11000
3	1-100	Male	510001-520000	11001-21000
4	1-100	Female	510001-520000	11001-21000
5	1-100	Female	510001-520000	1-10000
6	31-40	Male	520001-530000	11001-21000
7	31-40	Male	520001-530000	21001-31000
8	31-40	Male	520001-530000	11001-21000

The T is said to have identifying attribute like SSN and quasi identifiers (Q_{id}) which may reveal the identity of an individual when linked to external data. T may also have sensitive attributes (SA) which should not be disclosed to the public or may be disclosed after disassociating its values with an individual's other information. A few examples are Income and Disease. This paper considers both numerical and categorical types of sensitive values.

In k-Anonymity [1], if one record in the table has some value as Q_{id}, at least k-1 other records should have the value Q_{id} or the minimum group size on Q_{id} must be at least k. For example Table 2 is the 2-anonymized version of original Table 1.

Consider the equivalence class {Dancer, Person, [30-35]}. Since the sensitive attribute Disease = 'Hepatitis' for all members in the equivalence class, there is a privacy breach. This is termed as homogeneity attack [10]. To avoid this attack the Q_{id}-block which contains the sensitive values must be *l*-diversified. *l*-diversity provides privacy even when the data publisher does not know what kind of knowledge is possessed by the adversary. To satisfy the *l*-diversity in Table 2 we generalize Writer, Dancer and Female as Artist, Artist and Person respectively. Once generalization is done the attribute disease for the equivalence class {Artist, Person, [30-35]} will have diverse values as shown in the Table 3 thus satisfying *l*-diversity principle.

Wong et.al [3] proposed an anonymity model (α, k) to protect both identification and sensitive information to eliminate homogeneity attack. In (α, k), k portion is similar to k-anonymity and α is the maximum percentage of any sensitive value within any Q_{id}-block. Table 3 is (0.5, 2) - anonymous since $\alpha = 0.5$ for the set {job, sex, age} and sensitive value 'HIV'. There are two equivalence classes {t1, t2} and {t3, t4, t5, t6}. The first equivalence class has one tuple containing 'HIV' which implies that $\alpha = 0.5$. Similarly for the second equivalence class $\alpha = 0.25$. Hence $\alpha \leq 0.5$ for 'HIV'.

In this paper GUI based verification tool PRUDENT (**PR**ivacy **U**nscrambler for **D**isclosing Id**ENT**ity) is developed. The developed framework reveals the anonymity levels for anonymized (ΔT) dataset for a given dataset T. Any dataset that is being anonymized by using principles like k-anonymity, *l*-diversity and (α, k)- anonymity without any loss of generality by nature can be checked for anonymity levels using PRUDENT. This paper is divided into five sections. Section 2 deals with the related work. Section 3 shows the architecture of PRUDENT and analysis of the algorithm. Section 4 shows experimental results. In Section 5 a comparative study between Privacy FP- Tree and PRUDENT is done and Section 6 concludes the paper.

2 Related Work

To the best of our knowledge very less published work is available for verifying the anonymized dataset for privacy violations. Friedman et. al [8] developed the concept of k-anonymous decision tree in order to find the privacy breach. They initially scan the database and store the frequencies of all possible splits to determine whether the k-anonymity is breached

Sampson et al. [11] has worked on the concept of privacy FP-Tree which is an extension of FP-Tree to determine privacy violations for k-anonymity, *l*-diversity and (α, k) anonymity principles on an anonymized dataset. They store the dataset in a tree format and determine the anonymity levels by calculating the frequency of each identifier from the tree.

The construction of the Privacy FP-Tree is shown in Fig.1 for the anonymized dataset as shown in the Table 4. Each leaf node of the FP-Tree represents one unique quasi-identifier block of the dataset. The sensitive values are appended to each leaf node associated with the correct Q_{id}-block in the form of a linked list as shown in the Fig.1. Privacy FP-tree determines the k-anonymity value of the data set by finding the minimum number of rows that are located in each Q_{id}-block. For example the node

"510001-520000" has frequency of 3 which means that "510001-520000, 1-100, Male" is repeated thrice in the dataset. The minimum frequency is k for that dataset. To find *l*-diversity initially the unique sensitive values within each Q_{id}-block must be identified. This is achieved by finding the minimum depth of the linked list which is stored at the leaf node.

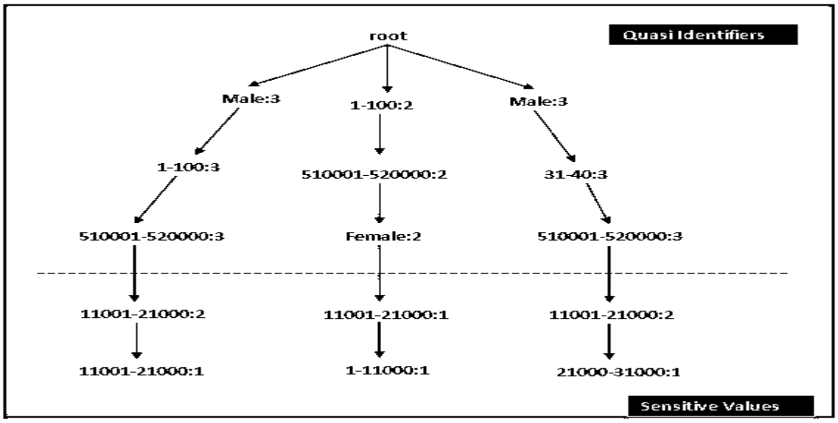


Fig. 1. Privacy FP-Tree

***Motivation for* PRUDENT**

Verifying whether the dataset has been correctly anonymized is required to assure the user. The data publisher anonymizes the huge microdata claiming that the anonymity is guaranteed to k-anonymous level. For an ordinary user or intermediary anonymity verification process is cumbersome. This motivation lead to the development of the PRUDENT, which is capable of checking k, *l* and (α, k) anonymity principles, and several others can be plugged into. Our algorithm uses simple itemset generation for determining the anonymity levels of aforementioned principles. Experimentations proved that PRUDENT is inexpensive when compared to the existing approaches.

3 The PRUDENT

In general FP-growth, Apriori [13] methods are used for mining frequent itemsets from a database using horizontal data format (Table 4) and is represented as {T_{id}: itemset} where T_{id} is the tuple-id and itemset is the set of attributes in the dataset. We can represent the same in vertical format [13] as {itemset: T_{id}-set} as shown in Table 5. PRUDENT (Fig.6) uses vertical format for generating frequent itemsets for detecting the anonymity levels of an anonymized dataset. The architecture of the PRUDENT is shown in Fig.2. In Section I the microdata T is given to the anonymizer. The anonymizer applies k, *l* and (α, k) anonymity principles over T and releases the anonymized dataset ΔT. ΔT is given as an input to Horizontal to Vertical data format Converter in section II. The itemset generator then generates item sets.

Later in section III k, *l* and α values are determined from the itemset with the help of k, *l* and α Finder. PRUDENT also generates a report showing the equivalence classes and their corresponding sensitive values. Due to space limit we omitted the snapshot of the report generator.

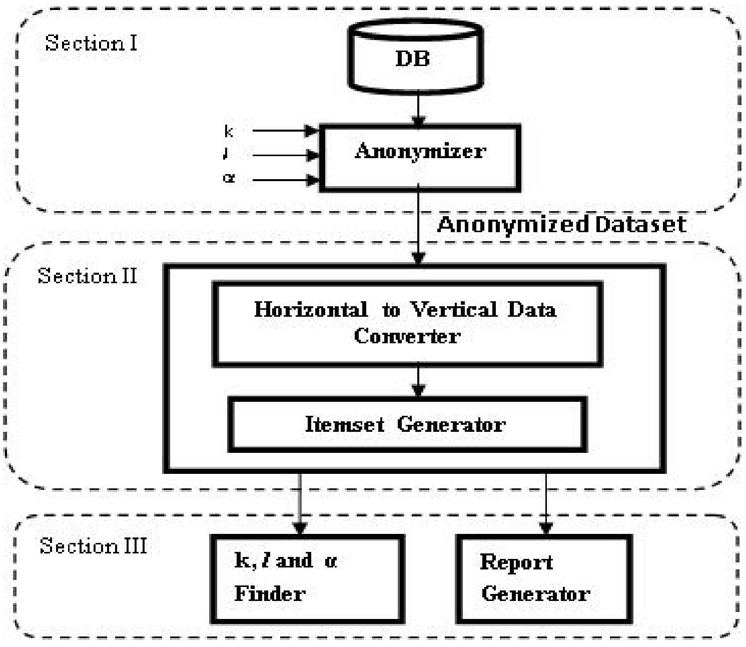


Fig. 2. Architecture of PRUDENT

3.1 Verification Algorithm

Vertical data set construction from an anonymized data set is shown in algorithm 1. Q_{id}-set generator in algorithm 2 generates itemsets in the form of {itemset: T_{id}-set} from the vertical dataset. Intersection operations are performed on the tuple-id sets (T_{id}) to produce T_{id+1} sets and union operations are applied on the corresponding Q_{id}-sets until $T_{id(k+1)}$ is not empty to form respective equivalence classes. Once all combinations of the Q_{id}-sets are generated, k, *l* and α are determined. The 1-itemset shown in Table 5 is the result produced after converting the horizontal dataset in Table 4 into vertical format.

Algorithm 2 is then applied on this 1-itemset to generate the subsequent itemsets. Consider the Q_{id}-sets {1-100} and {male}. When intersection operation is performed on their corresponding T_{id}-sets -[1,2,3,4,5] and [1,2,3,6,7,8] it produces Q_{id}-set {1-100, male} and the corresponding T_{id}-set is [1,2,3]. This process is repeated until all itemsets are produced. In some cases not all combinations of the Q_{id}-sets are produced since the intersections of T_{id}-sets will result in a null candidate set. Consider the 1-Itemset shown in Table 5. When intersections are performed on tuple-id sets [1,2,3,4,5] and [6,7,8] the resultant tuple-id set is a null candidate set. All such null candidate sets are eliminated in PRUDENT for optimal memory usage.

Algorithm 1. Vertical Dataset Constructor

```
Input: A anonymized dataset DS
Output: Vertical Dataset of anonymized dataset (VDS)
Method: The vertical dataset is constructed as follows
1. Begin
2.       Scan the anonymized dataset DS
3.       Create all items
4.       for each $T_i$ in DS do
5.          for each $Q_{id}$ in $T_i$ do
6.              Insert $T_{id}$ of $Q_{id}$ to $T_{id}$-set which Corresponding with $Q_{id}$
7.          end for
8.       end for
9. end Begin
```

Algorithm 2. Q_{id} sets generator

```
Input: A vertical dataset (VDS)
Output: The set of frequent patterns of $Q_{id}$'s with associated sensitive values and
        k, l and α values of the anonymized dataset
Method: The frequent patterns of $Q_{id}$ are constructed as follows
1. Begin
2.       for each $x_i$ in VDS do
3.          while the $I_{i+1}$ item in VDS <> the last item do
4.                //perform itemsets generation
5.             $TID_{x_{k+1}}=TID_{x_k} \cap TID_{I_k}$
6.             If ($TID_{x_{k+1}}$! = Emptyset) then
7.                $QID_{x_{k+1}}=QID_{x_k} \cup QID_{I_k}$
8.                Store $TID_{x_{k+1}}$ along with $QID_{x_{k+1}}$ to (k+1) itemsets
9.             end If
10.         end while
11.      end for
12.      Store all (k+1) itemsets into VDS
13.      Go to step (14) if all Qid set is generated otherwise Go to step 1 increment
           k value by 1
14.       Call k(VDS) //Finding k-value
15.       Call $l$(VDS) //Finding $l$-value
16.       Call α(VDS) //Finding α-Value
17. end Begin
```

3.2 Verifying for k-Anonymity

Let E= {E_1, E_2, E_3.. E_n} be the set of n equivalence classes of the dataset ΔT represented in the form of {Q_{id}-set: T_{id}-set} where T_{id}-set = { T_{id1}, T_{id2},….. T_{idn}} and E_i={ Q_{id}-set : T_{id}-set}. The size of ith equivalence class is k_i. $k_i = |E_i| = |T_{id}\text{-set}_i|$ where $1 \leq i \leq n$. The k- value for the given dataset is given by (1). Consider the 3-itemset as shown in the Table 7. Let the first, second and third items in the Table 7 be E_1, E_2 and E_3 respectively with values corresponding to 3, 2 and 3. The k-value for the dataset in Table 4 is Min (E_1, E_2, E_3) = 2.

$$k=\text{Min}\ \{\ |k_i|\ \}\ \text{where}\ 1\leq i\leq n \tag{1}$$

Table 5. 1-Itemset

Q_{id}-set	T_{id}-set
{1-100}	[1, 2, 3, 4, 5]
{31-40}	[6, 7, 8]
{male}	[1, 2, 3, 6, 7, 8]
{female}	[4, 5]
{510001-520000}	[1, 2, 3, 4, 5]
{520001-530000}	[6, 7, 8]

Table 6. 2- Itemset

Q_{id}-set	T_{id}-set
{1-100, male}	[1, 2, 3]
{1-100, female}	[4, 5]
{1-100,510001-520000}	[1, 2, 3, 4, 5]
{31-40, male}	[6, 7, 8]
{31-40, 520001-530000}	[6, 7, 8]

Table 7. 3- Itemset

Q_{id}-set	T_{id}-set
{1-100, male, 510001- 520000}	[1, 2, 3]
{1-100, female, 510001-520000}	[4, 5]
{31-40, male, 520001-530000}	[6, 7, 8]

Table 8. $T_{id's}$ and sensitive values

T_{Id}	Salary
1	11001-21000
2	1001-11000
3	11001-21000
4	11001-21000
5	1-10000
6	11001-21000
7	21000-31000
8	11001-21000

3.3 Verifying for *l*-Diversity

l-diversity verification is done by finding the value of *l*. Initially all the sensitive values are stored along with the tuple-id's before the itemsets are produced as shown in Table 8. Let $S_i =\{\ s_1, s_2,\ldots s_m\}$ be the set of sensitive values of the corresponding tuple-ids in the tuple-id set. Let $L_i = |\ \bigcup_{j=1}^{m} S_j\ |$ where $1\leq i\leq n$, where L_i is the number of distinct sensitive values in ith equivalence class. The *l*-value for the dataset ΔT is determined by (2).

$$l=\text{Min}\ \{L_i\}\ \text{where}\ 1\leq i\leq n \tag{2}$$

3.4 Verifying for (α, k) Anonymity

The (α, k) verification process is divided into two portions α and k. While k is verified as was done above, α can be determined by calculating the Max ($P_{E_i}(s_k)$), where $P_{E_i}(s_k)$ is the probability that the sensitive value s_k occurs equivalence class E_i and is defined in equation (3).

$$P_{E_i}(s_k)=\frac{|(E_i, s_k)|}{|E_i|} \quad (3)$$

where (E_i, s_k) is the set of tuples containing s_k in the given equivalence class E_i and $|(E_i, s_k)|$ is the cardinality of the set. For each equivalence class α is calculated as shown below.

$$\alpha E_i = \text{Max}\ \{\ P_{E_i}(s_k) : \forall\ s_k \in S\}\ \text{ where } \ 1 \leq k \leq |S| \quad (4)$$

$$\alpha = \text{Max}\{\ \alpha E_i\ \}\ \text{where } 1 \leq i \leq n \quad (5)$$

4 Experimentation

Experiments are performed on Intel Core2 Duo @ 2.93 GHz with 1024MB being allocated separately for the Netbeans platform. The experimentation was performed on the Adult dataset available at UCI Machine Learning Repository [15]. The dataset consists of age, sex and government attributes as Quasi Identifiers. Disease attribute was taken as sensitive attribute apart from Salary. We assumed that each equivalence class is of size 1,000 tuples. This can be changed if necessary.

The first part of the experimentations is confined to the conversion process of the dataset from horizontal to vertical format.

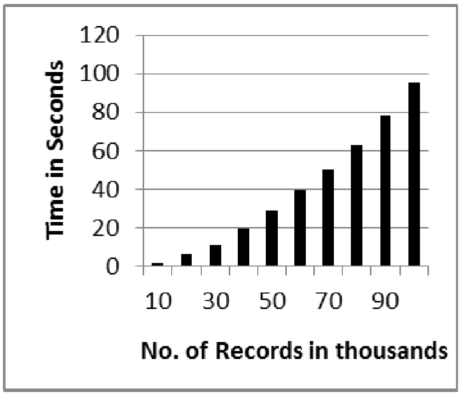


Fig. 3. Horizontal to Vertical Conversion

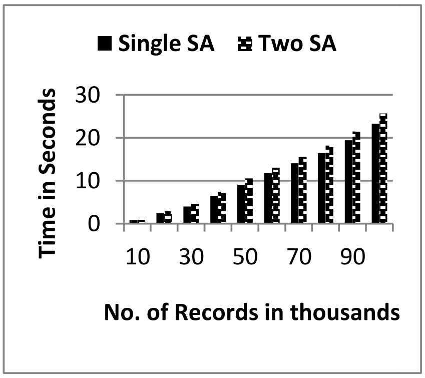


Fig. 4. Itemset Generation

The conversion process took less than 2 minutes as shown in Fig. 3 for the dataset size of 1, 00, 000 with 3-quasi and 2-sensitive attributes (SA). The itemset generation took less than 26 seconds for two sensitive attributes. Results show that PRUDENT is scalable as shown in Fig. 4 even for multiple sensitive attributes.

5 PRUDENT Vs. Privacy FP- Tree

The time complexity for constructing Privacy FP-Tree is O(nlogn) [11]. In our approach the time for constructing the itemset generation is also O(nlogn). The privacy properties of k-anonymity, *l*-diversity and (α, k)-anonymity are determined after the construction of Privacy FP-Tree [11] and itemset generation (Our Approach). The complexity for determining the privacy properties for the both approaches are explained and compared in the following sections.

5.1 k-Anonymity

Privacy FP-tree determines the k-value after tree construction [11]. The number of steps required for travelling all the leaf nodes is given by equation 6.

$$T=\frac{(q+1)(|E|+1)}{2}-1 \text{ where } |E|=\frac{N}{|E_i|} \tag{6}$$

T is average number of nodes traversed by the privacy FP-tree for finding k-anonymity. Let E be the set of equivalence classes E_i and N be the total number of tuples in the database. Let q be the number of quasi identifiers. The time complexity for performing these operations is O(T). In PRUDENT there is no necessity of travelling all the leaf nodes. In our approach we simply perform itemset generations (Section 3.2) and the k-value is calculated as per equation (1) mentioned in section 3.3. The time complexity for doing such operation is O(E).

5.2 *l*–Diversity

Once the final itemset was generated we retrieve the sensitive values corresponding to the tuple-id's within each equivalence class with n sensitive values. The retrieved sensitive values are sorted out using merge sort where the complexity is $nlogn$ and n comparisons are required to compute distinct sensitive values in each equivalence class. We perform all these operations in parallel. So, the time complexity is O(S). For finding the *l*-diversity of a dataset initially the privacy FP- Tree must be constructed. During the tree construction, the sensitive values are appended to the leaf node of the tree using linked list format. First one sensitive value is appended. When the second sensitive value is added there must be a comparison with the pervious sensitive value. If the prior and the later are same we increment the count value of the prior sensitive value or else a new node is added for the prior one.

Let us assume that |E| equivalence classes are formed and for each equivalence class let the sensitive values are m_i. Two cases were identified: if all the sensitive values are distinct the number of comparisons will be $\sum_{i=0}^{|E|}\frac{m_i(m_i-1)}{2}$. If all the sensitive values are similar the number of comparisons will be $\sum_i^E m_i$. On an average the complexity is O(m^2) for constructing the linked list of sensitive values.

$$S=\frac{|E|(mlogm+m)}{2} \tag{7}$$

Before checking for the *l*-value, we must traverse till the leaf node whose order of complexity will be O(T) and for finding the depth of the linked list the complexity is O(m) for each equivalence class. So the overall complexity for travelling all the |E| equivalence classes is $(O(T)+O(m^2)+O(m))$. Hence, the total complexity is O $(|E|m^2)$.The performance of the PRUDENT when compared to Privacy FP-Tree for k-anonymity is encouraging. Fig. 5(a), 5(b), 5(c) show how the performance varies for different $Q_{id's}$. Fig. 5(d) shows the average comparisons for *l*-diversity which are enormously increased for Privacy FP-Tree when compared with our approach section.

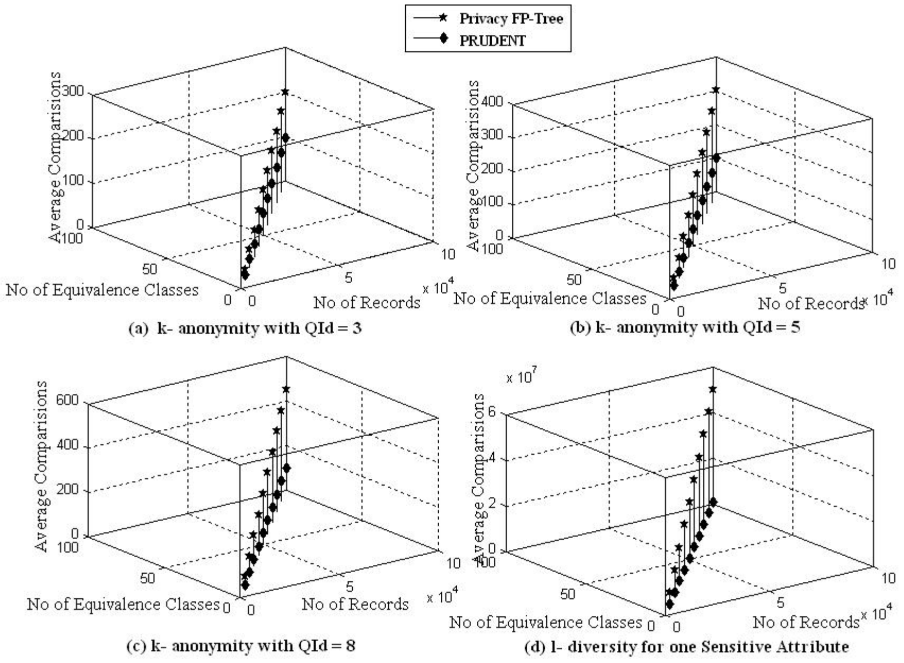


Fig. 5. Comparisons Privacy FP-Tree Vs PRUDENT

5.3 (α, k) – Anonymity

In (α, k) - anonymity the k-portion's complexity is similar to the complexity of k-anonymity, O(E) and the α portions complexity is similar to the complexity of *l*-diversity, O(S). So the overall complexity is $O(E) + O(S) \approx O(S)$.

6 Conclusions

PRUDENT, a tool to verify the anonymity levels for a given anonymized dataset is discussed. The tool gives the flexibility of selecting the database, the quasi-identifiers and sensitive attributes. The tool will come in handy for both users and publishers. It allows the selection of different combinations of the quasi-identifier attributes.

The itemset (Equivalence classes) determines k, *l* and α values for a given anonymized dataset. PRUDENT checks any kind of anonymized dataset which is anonymized by using any anonymized framework like generalization, suppression etc. This paper discusses an approach for verifying anonymity levels for a given anonymized dataset. The tool presented works for all the datasets that are anonymized using k-anonymity, *l*-diversity, (α, k) anonymity principles using the proposed algorithm. A comparison is also made with an existing method based on Privacy FP-Tree.

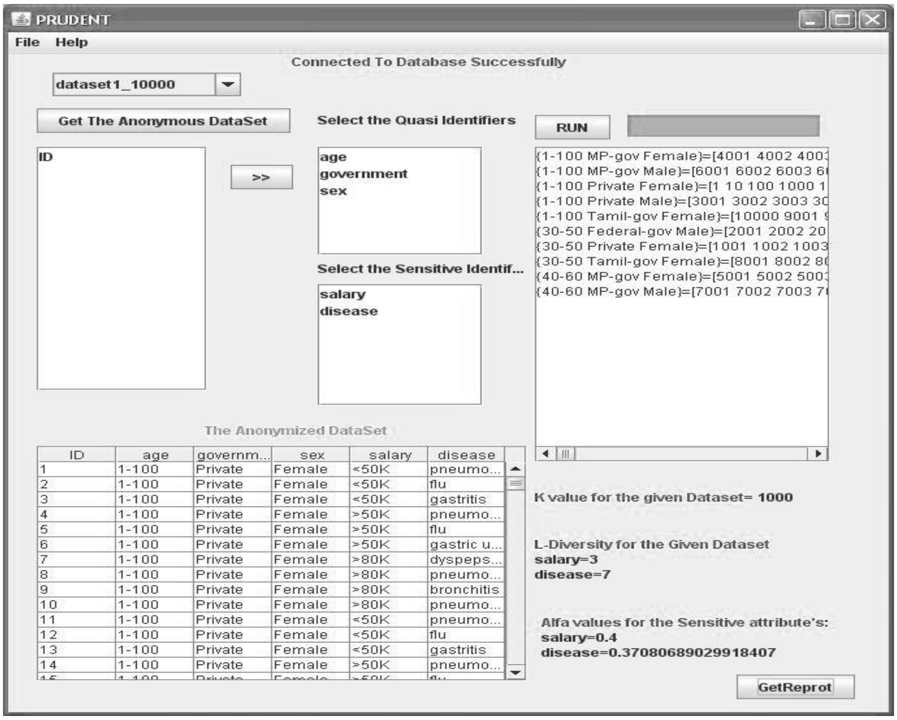
Acknowledgements

This work was supported by Grant SR/S3/EECE/0040/2009 from Department of Science and Technology (DST), Government of India. We would like to thank the anonymous reviewers for their insightful comments.

References

1. Sweeney, L.: K-anonymity: a model for protecting privacy. International Journal on Uncertainty 10(5), 557–570 (2002)
2. Narayanan, A., Shmatikov, V.: Robust De-anonymization of Large Datasets (February 5, 2008)
3. Wong, R., Li, J., Fu, A., Wang, K.: (α, k) Anonymity: An Enhanced k-Anonymity Model for Privacy Preserving Data Publishing. In: KDD (2006)
4. Dwork: An Ad Omnia Approach to Defining and Achieving Private Data Analysis. In: Proceedings of the First SIGKDD International Workshop on Privacy, Security, and Trust in KDD (2007)
5. Sweeney, L.: Weaving technology and policy together to maintain confidentiality. J. of Law, Medicine and Ethics 25(2-3), 98–110 (1997)
6. Liew, K., Choi, U.J., Liew, C.J.: A data distortion by probability distribution. ACM TODS 10(3), 395–411 (1985)
7. Xiao, X., Tao, Y.: Personalized Privacy Preservation. In: SIGMOD (2006)
8. Friedman, R.W., Schuster, A.: Providing k-anonymity in data mining. The VLDB Journal, 789–804 (2008)
9. Wang, K., Fung, B.C.M., Yu, P.S.: Handicapping Attacker's Confidence: An Alternative to k-Anonymization. Knowledge and Information Systems: J, KAIS (2006)
10. Machanavajjhala, J.G., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: Proc. 22nd Intl. conf. Data Engg (ICDE), p. 24 (2006)
11. Sampson, P., Barker, K.: Privacy FP-Tree. In: Chen, L., et al. (eds.) DASFAA 2009 Workshops. LNCS, vol. 5677, pp. 246–260 (2009)
12. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and Suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 571–588 (2002)
13. Han, J., Kamber, M.: Data Mining Concepts and Techniques. Morgan Kaufmann, San Fransisco (2006)
14. Fung, B.C.M., Wang, K.E., Chen, R., Yu, P.S.: Privacy- Preserving Data Publishing: A Survey on Record Developments. ACM Computing Surveys 42(4), Article 14 (2010)
15. UCI Repository of Machine Learning databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html`
16. Hansell, S.: AOL removes search data on vast group of web users. New York Times (August 8, 2006)

Appendix:



Connected To Database Successfully

dataset1_10000

ID

Select the Quasi Identifiers

age
government
sex

Select the Sensitive Identif...

salary
disease

```
{1-100 MP-gov Female}=[4001 4002 400
{1-100 MP-gov Male}=[6001 6002 6003 6
{1-100 Private Female}=[1 10 100 1000 1
{1-100 Private Male}=[3001 3002 3003 30
{1-100 Tamil-gov Female}=[10000 9001
{30-50 Federal-gov Male}=[2001 2002 20
{30-50 Private Female}=[1001 1002 1003
{30-50 Tamil-gov Female}=[8001 8002 8
{40-60 MP-gov Female}=[5001 5002 500
{40-60 MP-gov Male}=[7001 7002 7003 7
```

The Anonymized DataSet

ID	age	governm...	sex	salary	disease
1	1-100	Private	Female	<50K	pneumo...
2	1-100	Private	Female	<50K	flu
3	1-100	Private	Female	<50K	gastritis
4	1-100	Private	Female	>50K	pneumo...
5	1-100	Private	Female	>50K	flu
6	1-100	Private	Female	>50K	gastric u...
7	1-100	Private	Female	>80K	dyspeps...
8	1-100	Private	Female	>80K	pneumo...
9	1-100	Private	Female	>80K	bronchitis
10	1-100	Private	Female	>80K	pneumo...
11	1-100	Private	Female	<50K	pneumo...
12	1-100	Private	Female	<50K	flu
13	1-100	Private	Female	<50K	gastritis
14	1-100	Private	Female	>50K	pneumo...

K value for the given Dataset= **1000**

L-Diversity for the Given Dataset
salary=3
disease=7

Alfa values for the Sensitive attribute's:
salary=0.4
disease=0.37080689029918407

Fig. 6. PRUDENT

Chaos Based Image Encryption Scheme Based on Enhanced Logistic Map

I. Shatheesh Sam[1], P. Devaraj[2], and R.S. Bhuvaneswaran[1]

[1] Ramanujan Computing Centre, College of Engineering, Guindy, Anna University, Chennai, India
[2] Department of Mathematics, College of Engineering, Guindy, Anna University, Chennai, India
shatheeshsam@yahoo.com

Abstract. Image encryption schemes are important to ensure the security during transmission or storage. In this paper, chaos based image encryption scheme based on enhanced logistic map is proposed. The first stage consists of row and column rotation and XORing with first chaotic key. In the diffusion process, the pixel values are altered sequentially so that the changes made to a particular pixel depends on the accumulated effect of all the previous pixels. The operations include nonlinear diffusion using the second chaotic key and alternative zig-zag diffusion of adjacent pixels and XORing with the third chaotic key. The number of rounds in the steps are controlled by combination of pseudo random sequence and original image. The security and performance of the proposed image encryption technique have been analysed using statistical analysis, key space analysis, differential analysis and entropy analysis. The scheme possesses good performance in encryption speed and is suitable for real-time image encryption and transmission.

Keywords: Enhanced Chaotic Map, Nonlinear Diffusion, Alternative zig-zag Diffusion.

1 Introduction

The degree to which individuals appreciate privacy differ from one person to another. Various methods have been investigated and developed to protect personal privacy. Security of multimedia is the most obvious one. In the last two decades increasing efforts have been made to use chaotic systems for enhancing some features of communications systems. The highly unpredictable and random look nature of chaotic signals is the most attractive feature of deterministic chaotic systems that may lead to novel applications.

Security of image data is receiving more attention due to the widespread transmission over various communication networks. It has been noticed that the traditional text encryption schemes fail to safely protect image data due to some special properties of these data and some specific requirements of image processing systems, such as bulky size and strong redundancy of uncompressed data.

Therefore, designing good image encryption schemes has become a focal research topic since the early 1990s. Inspired by the subtle similarity between chaos and cryptography, a large number of chaos-based image encryption schemes has been proposed [2–5]. Most chaotic image encryptions or encryption systems use the permutation substitution architecture. These two processes are repeated for several rounds, to obtain the final encrypted image. Fridrich [1] suggested a chaotic image encryption method composed of permutation and substitution. All the pixels are moved using a 2D chaotic map. Sam et.al. [12] used odd keys in the permutation stage, byte susbtitution and imporved chaotic maps in the diffusion stage. The new pixels moved to the current position are taken as a permutation of the original pixels. In the substitution process, the pixel values are altered sequentially. Unfortunately, many of these schemes have been found insecure, especially against known and/or chosen-plaintext attacks. The one dimensional chaos system has the advantages of simplicity and high security. Some of the cryptanalysis techniques [7-11] are suggested to break the scheme and reduce the flaws in the algorithm design.

In this paper, chaos based image encryption scheme based on enhanced logistic map is suggested to overcome the weakness of security level. The algorithm uses significant features such as sensitivity to initial condition, permutation of keys, enhanced chaotic maps, nonlinear diffusion and alternative zig-zag diffusion. The nonlinearity is used to overcome the limitation of the other schemes. The rest of this paper is organized as follows. Section 2 introduces the logistic map and enhanced maps. In section 3, the image encryption based on enhanced logistic map is proposed including a new scheme. Section 4, analyses the security of new algorithm. Finally, the conclusions are discussed in section 5.

2 Logistic Map

Logistic map is a simple but broadly researched dynamic system. A classical logistic map is defined by

$$x_{i+1} = ax_i(1-x_i)$$

where a is system parameter, $0 < a \leq 4$ and x_i is a floating number in (0,1), $i = 0, 1, 2, 3 \ldots$. When $a > 3.568945672$, this system become chaotic in behavior, in other words, the sequence $\{x_i, i = 0, 1, 2, \ldots\}$ produced based on logistic map with the initial value x_0 is neither periodic nor convergent. A graphical way to visualize this phenomenon is as shown in Fig. 1, where the bifurcation map brings information about the dynamics of the system.

Since not all the fixed points for a one dimensional mapping properties are identical and a feature that makes the difference is their stability, a way to determine it, is by calculating the lyapunov exponent where a positive value indicates instability and for chaotic systems the larger value is better. The basic expression of the discrete lyapunov exponents is defined as

$$\lambda_F = \frac{1}{M} \sum_{i=0}^{M-1} ln \frac{d(F(m_{i+1}), F(m_j))}{d(m_{i+1}, m_j)}$$

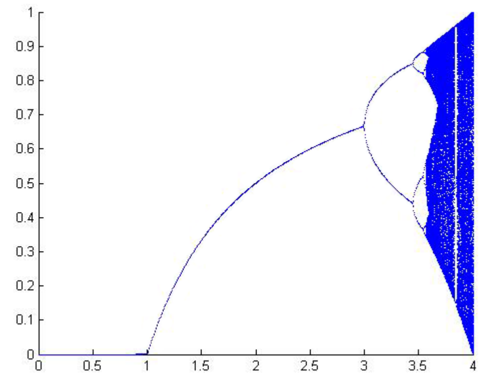


Fig. 1. Bifurcation diagram for the logistic map

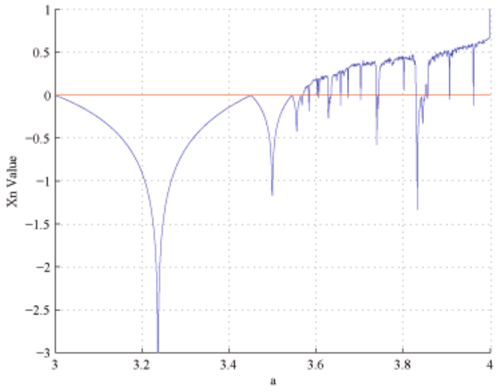


Fig. 2. Lyapunov exponent for the logistic map

where m_i is the subset of trajectory of a digitalized map F in length M and $d(m_{i,}, m_j)$ is the distance between $m_i$and m_j. The computation of the logistic map is shown in Fig. 2.

Though, the logistic map is better for image encryption which has some common problems such as stable windows, blank windows, uneven distribution of sequences and weak key[6]. New types of enhanced logistic maps have been proposed in the paper to alleviate the problems in the logistic map. The maps are mixed together so as to achieve larger key space and to attain chaotic behavior. We have attempted to improve it by chaotic transformation. The proposed enhanced chaotic logistic maps are defined and keys are generated in the section. Thus, the proposed enhanced chaotic logistic map does not have security issues which are present in the logistic map. Moreover, the resulting chaotic sequences are uniformly distributed (see the Figure 3) and the key size has been increased greatly.

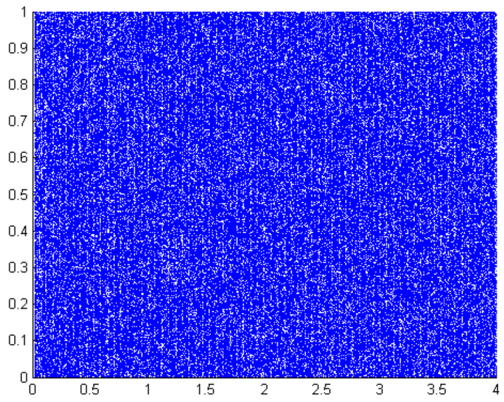


Fig. 3. Distribution of sequence for the enhanced map

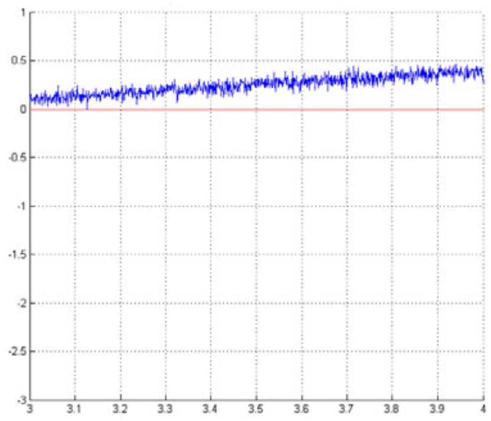


Fig. 4. Lyapunov exponent for the enhanced map

The lyapunov exponent computation of the enhanced map is shown in Fig. 4. Lyapunov exponent is bigger than zero, the system is chaotic.

3 Proposed Scheme

The architecture of the proposed substitution-diffusion based cryptosystem is shown in Fig. 5. The scheme consists of three major phases, rotation, nonlinear diffusion and alternative zig-zag diffusion. In the confusion stage, both the rotation on pixel position, the change of pixel value and XORing with chaotic key are carried out at the same time while the diffusion process remains unchanged. As a result, the pixel value mixing effect of the whole cryptosystem is contributed by confusion and diffusion operations: the modified confusion process, nonlinear diffusion and the alternative zig-zag diffusion function.

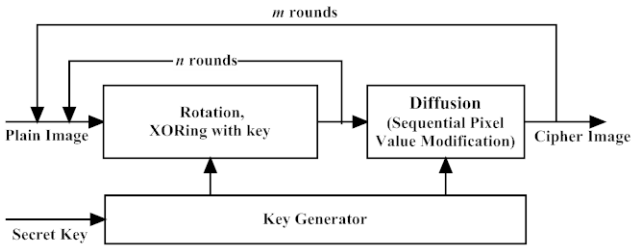

m rounds
n rounds
Plain Image
Rotation, XORing with key
Diffusion (Sequential Pixel Value Modification)
Cipher Image
Secret Key
Key Generator

Fig. 5. Proposed architecture

The plain image is stored in a two dimensional array of pixels. In this, $1 \leq i \leq H$ and $1 \leq j \leq W$, where H and W represent height and width of the plain image in pixels.

3.1 Key Generation

The enhanced chaotic map and the keys have been generated in the following way:

```
$for\, i = 1\, to\, 256$
    $for\, j = 1\, to\, 256$
        $x_{i,j+1} = (3.853429 \times k_1 \times (1 - x_{i,j}) + (y_{i,j})^2)\, mod\, 1$
        $y_{i,j+1} = (3.979283 \times k_2 \times y_{i,j} \times (1/1 + (x_{i,j+1})))\, mod\, 1$
        $z_{i,j+1} = (3.769943 \times k_3 \times (x_{i,j+1})^2 \times y_{i,j+1} \times sin(z_{i,j}))\, mod\, 1$
        $KX_{i,j} = \lfloor x_{i,j+1} \times 256 \rfloor$
        $KY_{i,j} = \lfloor y_{i,j+1} \times 256 \rfloor$
        $KZ_{i,j} = \lfloor z_{i,j+1} \times 256 \rfloor$
    $end$
    $x_{i+1,1} = x_{i,j+1}$
    $x_{i+1,1} = x_{i,j+1}$
    $z_{i+1,1} = z_{i,j+1}$
$end$
```

where $| k_1 |> 31.5$, $| k_2 |> 25.7$, $| k_3 |> 21.3$ respectively. To increase the key size we can use k_1, k_2, k_3 as another set of keys. Along with the key k_i the distribution of the sequences becomes better. $KX_{i,j}, KY_{i,j}, KZ_{i,j}$ are the set of chaotic keys.

3.2 Rotation and XORing

The image is rotated by rotation function using first chaotic key and the resultant values are XORed with same chaotic key. All rotation are done based on the key values.

The rotation functions are described as follows:

```
$for\, i = 1\, to\, H$
    $for\, j = 1\, to\, W$
```

```
        $C[i,j] = R_{(i,j+KX_{i,j}) mod 256} \oplus KX_{i,j}$
    $end$
$end$
```

where KX is the first chaotic key. R is to rotate each value in the i^{th} row and j^{th} column of the image r times.

3.3 Nonlinear Diffusion

Diffusion refers to the property that redundancy in the statistics of the plain text is dissipated in the statistics of the cipher text. The diffusion is obtained by 4 Least Significant Bits (LSB) circular shift method using randomly chosen image. The resultant values are again XORed with second chaotic key. The procedure for the nonlinear diffusion is as follows:

```
$for\, i = 1\, to\, H$
    $for\, j = 1\, to\, W$
        $C_{i,j} = (C_{i,j} \ggg 4)\, mod\, 256$
        $D_{i,j} = C_{i,j} \oplus KY_{i,j}$
    $end$
$end$
```

where $KY_{i,j}$ is the second chaotic key. $D[i,j]$ denotes the $(i,j)^{th}$ pixel of the cipher image. The combination of 4 bit circular shift and XORing make the encryption operation nonlinear and hence the system becomes strong against known/chosen plaintext attack.

3.4 Alternative zig-zag Diffusion

In this, we read the values in the alternative zig-zag (see Figure 6) manner as follows: $R_{11}, R_{21}, R_{22}, R_{12}, R_{13}, R_{23}, R_{33}, R_{32}, R_{31}, R_{41}$ etc. However, the diffusion is obtained with the help of alternative zig-zag XORing and XORing with third chaotic key. Therefore, these operations enhance diffusion property and hence improves the security features.

The procedure for alternative zig-zag diffusion is as follows:

$$\begin{aligned}
&R_{11} = R_{11} \oplus z_{11},\\
&R_{21} = R_{11} \oplus R_{21} \oplus KZ_{12},\\
&R_{22} = R_{21} \oplus R_{22} \oplus KZ_{21},\\
&R_{12} = R_{12} \oplus R_{22} \oplus KZ_{31},\\
&R_{13} = R_{13} \oplus R_{12} \oplus KZ_{22},\\
&R_{23} = R_{23} \oplus R_{13} \oplus KZ_{13},\\
&R_{33} = R_{33} \oplus R_{23} \oplus KZ_{14},\\
&\dots\dots\dots\dots\\
&\dots\dots\dots
\end{aligned}$$

where KZ is the third chaotic key. The above procedure is continued till the last pixel is reached.

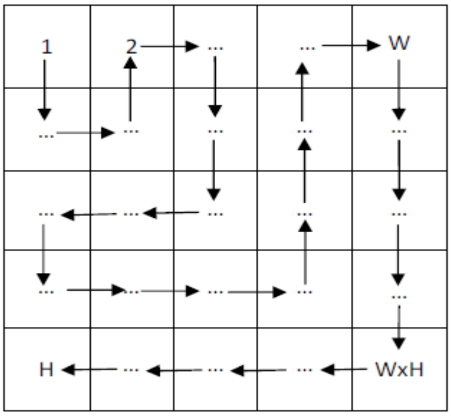


Fig. 6. Alternative zig-zag pixel value reading

4 Performance and Security Analysis

We have made several expriments to check the security of the proposed cryptosystem. Statistical tests include histogram analysis, calculation of the correlation coefficients of adjacent pixels. Security tests against differential attack include calculation of the NPCR and UACI, and information entropy evaluation.

4.1 Histogram Analysis

Histograms may reflect the distribution information of the pixel values of an image. An attacker can analyse the histograms of an encrypted image by using some attacking algorithms to get some useful information of the original image. Thus, the histograms of an encrypted image should be as smooth and evenly distributed as possible, and should be very different from that of the plaintexts. Fig. 7. shows a comparison of the histograms between plaintext and encrypted images.

4.2 Statistical Analysis

Randomly select thousand pairs of adjacent pixels in vertical, horizontal and diagonal directions from the plain image and ciphered image and calculate the correlation coefficients of two adjacent pixels according to the following formula

$$r_{xy} = \frac{cov(x,y)}{\sqrt{D(\alpha)}\sqrt{D(\beta)}}$$

where

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)),$$

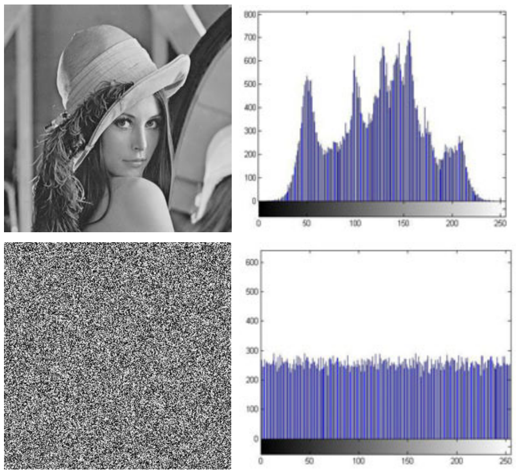


Fig. 7. Histogram analysis of plain image and cipher image

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2$$

where x and y denote two adjacent pixels and N is the total number of duplets (x, y) obtained from the image.

Table 1. Correlation coefficients of plain image and ciphered image

Correlation	Vertical	Horizontal	Diagonal
Plain image	0.9856	0.9821	0.9687
Ciphered image	0.0067	0.0043	0.0021

Table 1 shows the results of correlation coefficients of two adjacent pixels. The result indicates that the correlation of two adjacent pixels of the plain image is significant, while that of the ciphered image is very small. Thus, it is clear that the proposed algorithm is robust.

4.3 Differential Analysis

Differential attack would become ineffective even if a single pixel change in the plain-image causes a significant difference in the cipher-image. In order to measure this capability quantitatively, the following measures are usually used: number of pixels change rate (NPCR) and unified average changing intensity (UACI).

They are defined as follows:

$$D_{ij} = \begin{cases} 1, & if\, C_{ij} \neq C'_{ij} \\ 0, & otherwise \end{cases}$$

The NPCR is defined as

$$NPCR = \frac{\sum_{i,j} D_{ij}}{M \times N} \times 100\%$$

The UACI is defined as

$$UACI = \frac{1}{M \times N} \left[\sum_{i,j} \frac{C_{ij} - C'_{ij}}{M \times N} \right] \times 100\%$$

where C_{ij} and C'_{ij} are the two cipher-images at position (i, j) whose corresponding plain-images have only one-pixel difference and M and N are the number of rows and columns of images. The results of NPCR and UACI are listed in Table 2.

Table 2. Sensitivity to ciphertext

	NPCR%	UACI%
Lena	99.6311	33.4989
Baboon	99.6334	33.4813
House	99.6341	33.4809
Tree	99.6331	33.4814

In order to assess the influence of changing a single pixel in the original image on the encrypted image, the $NPCR$ and the $UACR$ are computed in the proposed scheme. It is found that the NPCR is over 99.3% and the UACI is over 33.4%. The results show that a small change in the original image will result in a significant difference in the cipherimage, so the scheme proposed has good in anti differential attack.

4.4 Key Space Analysis

There are series of enhanced maps parameters, initial values and k_i values that can be used as key in our scheme. The key space is as large as the range between 220 to 420 bits. The key space is large enough to resist the attacks.

4.5 Avalanche Criterion

A small change in either the key or the plaintext should cause a drastic change in the ciphertext, ideally 50% difference in the bits of the cipher. The analysis is exhibited the changing rate of bits about 49.96%. So proposed scheme is nearly ideal.

4.6 Information Entropy Analysis

Information entropy is the most important feature of randomness. The entropy $H(s)$ of a message s can be calculated by

$$H(s) = \sum_{i=0}^{2^N-1} p(s_i) log \frac{1}{p(s_i)}$$

where $p(s_i)$ represents the probability of symbol s_i . In theory, a true random source should generate 2^8 symbols with equal probability, and the entropy is $H(s) = 8$. After calculation, we found the average entropy of the 'Lena' cipher image of this algorithm is about 7.9989 which is very close to the theoretical value $N = 8$.

Table 3. Entropy analysis of the proposed and other schemes

Cipher	Proposed	RC5	RC6
Lena	7.9989	7.9813	7.9839
Lion	7.9991	7.9863	7.9872

As shown in Table 3, we notice that the values obtained of our scheme are very close to the theoretical value of 8 than other schemes. Therefore the proposed algorithm is robust against the entropy attack.

5 Conclusion

In this paper, chaos based image encryption scheme based on enhanced logistic map is proposed. The proposed cipher provides good confusion and diffusion properties that ensure extremely high security. Confusion and diffusion have been achieved using rotation, nonlinear diffusion and alternative zig-zag diffusion. This scheme is immune to various types of cryptographic attacks like known/chosen plain text attacks and brute force attacks. We have carried out statistical analysis, key space analysis differential analysis and entropy analysis to demonstrate the security of the new image encryption procedure. Based on the various analyses, it is shown that the proposed scheme is more secure and fast and so more suitable for real time image encryption for transmission applications.

Acknowledgment

The first author is partly funded by All India Council for Technical Education (AICTE), New Delhi, India.

References

1. Fridrich, J.: Symmetric ciphers based on two-dimensional chaotic maps. Int. J. Bifurc. Chaos. 8(6), 1259–1284 (1998)
2. Tong, X., Cui, M.: Image encryption with compound chaotic sequence cipher shifting dynamically. Image Vision Comput. 26, 843–850 (2008)
3. Wong, K.W., Wok, B.S.H., Law, W.S.: A fast image encryption scheme based on chaotic standard map. Phys. Lett. A 372, 2645–2652 (2008)
4. Patidar, V., Pareek, N.K., Sud, K.K.: A new substitution diffusion based image cipher using chaotic standard and logistic maps. Commun. Nonlinear Sci. Numer. Simulat. 14, 3056–3075 (2009)
5. Patidar, V., Pareek, N.K., Purohit, G., Sud, K.K.: Modified substitution–diffusion image cipher using chaotic standard and logistic maps. Commun. Nonlinear Sci. Numer. Simulat. 15(10), 2755–2765 (2010)
6. Jianquan, X., Chunhua, Y., Qing, X., Lijun, T.: An Encryption Algorithm Based on Transformed Logistic Map. In: IEEE International Conference on Network Security, Wireless Communications and Trusted Computing, pp. 111–114 (2009)
7. Rhouma, R., Solak, E., Belghith, S.: Cryptanalysis of a new substitution-diffusion based image cipher. Commun. Nonlinear Sci. Numer. Simulat. 15, 1887–1892 (2010)
8. Alvarez, G., Shujun, L.: Cryptanalyzing a nonlinear chaotic algorithm (NCA) for image encryption. Commun. Nonlinear Sci. Numer. Simulat. 14, 3743–3749 (2009)
9. Li, C., Li, S., Alvarez, G., Chen, G., Lo, K.-T.: Cryptanalysis of two chaotic encryption schemes based on circular bit shift and XOR operations. Phys. Lett. A 369(1-2), 23–30 (2007)
10. Li, C., Li, S., Asim, M., Nunez, J., Alvarez, G., Chen, G.: On the security defects of an image encryption scheme. Image Vis. Comput. 27, 1371–1381 (2009)
11. Li, C., Shujun, L., Chen, G., Halang, W.A.: Cryptanalysis of an image encryption scheme based on a compound chaotic sequence. Image Vis. Comput. 27, 1035–1039 (2009)
12. Sam, I.S., Devaraj, P., Bhuvaneswaran, R.S.: Enhanced Substitution-Diffusion Based Image Cipher Using Improved Chaotic Map. In: Das, V.V., Vijaykumar, R. (eds.) ICT 2010. CCIS, vol. 101, pp. 116–123. Springer, Heidelberg (2010)

Matrix Insertion-Deletion Systems for Bio-Molecular Structures

Lakshmanan Kuppusamy[1], Anand Mahendran[1], and Shankara Narayanan Krishna[2]

[1] School of Computing Science and Engineering, VIT University, Vellore-632 014, India
klakshma@vit.ac.in, manand@vit.ac.in

[2] Department of Computer Science and Engineering, IIT Bombay, Powai - 400 076, India
krishnas@cse.iitb.ac.in

Abstract. Insertion and deletion are considered to be the basic operations in Biology, more specifically in DNA processing and RNA editing. Based on these evolutionary transformations, a computing model has been formulated in formal language theory known as *insertion-deletion* systems. Since the biological macromolecules can be viewed as symbols, the gene sequences can be represented as strings. This suggests that the molecular representations can be theoretically analyzed if a biologically inspired computing model recognizes various bio-molecular structures like *pseudoknot, hairpin, stem and loop, cloverleaf* and *dumbbell.* In this paper, we introduce a simple grammar system that encompasses many bio-molecular structures including the above mentioned structures. This new grammar system is based on insertion-deletion and *matrix* grammar systems and is called *Matrix insertion-deletion grammars*. Finally, we discuss how the ambiguity levels defined for insertion-deletion grammar systems can be realized in bio-molecular structures, thus the ambiguity issues in gene sequences can be studied in terms of grammar systems.

Keywords: bio-molecules, structure representation, insertion-deletion systems, matrix grammars, ambiguity.

1 Introduction

In the last three decades, biology played a great role in the field of formal languages by being the root for the development of various biologically inspired computing models such as *sticker systems, splicing systems, Watson-Crick automata, insertion-deletion systems, p systems* [3], [8], [9]. Since, most of the language generating devices are based on the operation of rewriting systems, the insertion-deletion systems opened a particular attention in the field of formal languages. Informally, the insertion and deletion operations of an insertion-deletion system is defined as follows: If a string α is inserted between two parts w_1 and w_2 of a string w_1w_2 to get $w_1\alpha w_2$, we call the operation as insertion, whereas if a substring β is deleted from a string $w_1\beta w_2$ to get w_1w_2, we call the operation as deletion.

DNA molecules may be considered as strings over alphabet consisting of four symbols namely a, t, g and c. Similarly, RNA molecules may be considered as strings over alphabet consisting of four symbols namely a, u, g and c. Since the bio-molecular structures can be defined in terms of sequence of symbols (i.e., strings) there exists a correlation between formal grammars and bio-molecular structures. The following example shares a common point between formal grammar and molecular strings. Consider the following gene sequence $S = ctatcgcgatag$. As $\bar{a} = t$, $\bar{t} = a$, $\bar{g} = c$ and $\bar{c} = g$, the above gene sequence resembles the context-free (palindrome) language $\{w\bar{w}^R \mid w \in \{a, b\}^*\}$.

The gene sequences in the bio-molecular structures can be viewed as strings which has some common patterns in it. In [7], [6] it has been shown that there exists a relevance between the gene sequences and natural language constructs such as *triple agreements* : $L_1 = \{a^nb^nc^n \mid n \geq 1\}$, *crossed dependencies:* $L_2 = \{a^nb^mc^nd^m \mid n, m \geq 1\}$. We discuss in brief some of the important structures seen in bio-molecules. Fig.1. shows the two structures which are predominantly available in bio-molecules like proteins, DNA and RNA molecular structures. Fig.2. shows the coherence between natural language constructs and gene sequences. Fig.2.(a) and (b) represents the grammar oriented derivations of idealized RNA structures (# denotes empty string) whereas Fig.2.(c) represents a biological sequence which has a crossed dependency pattern. The corresponding languages for the above structures can be given as *hairpin language* (for Fig.2(a)), orthodox language (for Fig.2(b)) and pseudoknot structure (for Fig.2(c)). The formal language notations for such structures and for a few other structures are discussed in detail in the coming sections. For more details on Genome structures we refer to [2].

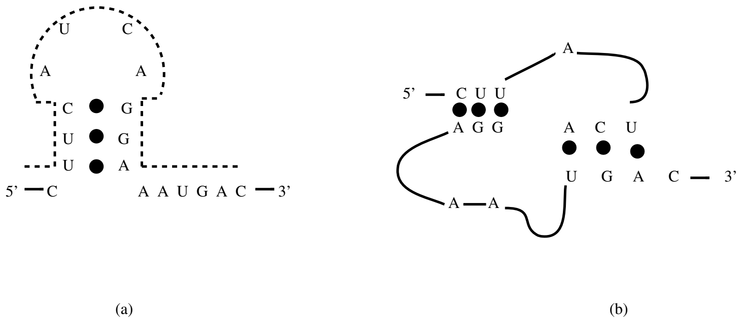


Fig. 1. Bio-molecular structures: (a)stem and loop (b)pseudo-knot

In the last two decades, many attempts have been made to establish the linguistic behaviour of biological sequences by defining new grammar formalisms like *cut grammars* [4], [5], [6], *crossed-interaction grammar* [7], *simple linear tree adjoining grammars* and *extended simple linear tree adjoining grammars* [15]

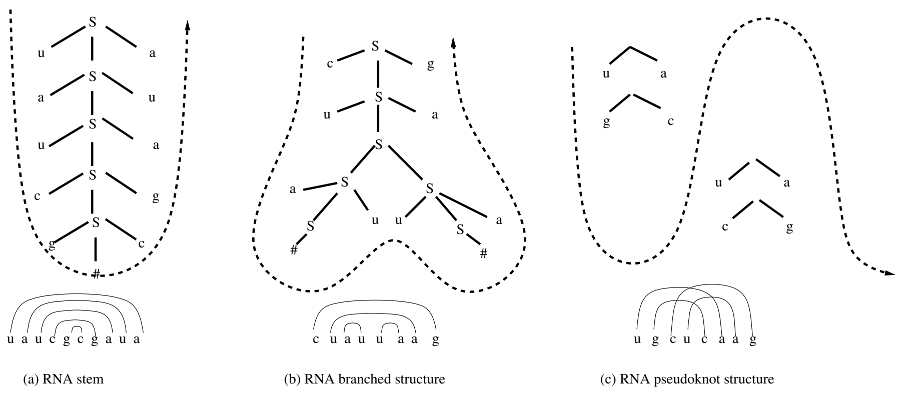


Fig. 2. RNA structures

which are capable of generating some of the biological structures mentioned above. However, there is no unique grammar system that encapsulate all discussed bio-molecular structures. This motivates us to introduce a simple and powerful grammar systems that captures all the essential and important bio-molecular structures.

Ambiguity is considered as one of the fundamental problems in formal language theory. A grammar is said to be ambiguous, if there exists more than one distinct derivation of the words in the generated language. As the insertion-deletion system can be applied in DNA processing [8], the ambiguity in DNA processing (which uses the insertion system) can happen in the following manner. Let W_1W_2 be a DNA strand and suppose we want to insert $W_3W_4W_5$ between W_1 and W_2 to obtain another DNA strand $W_1W_3W_4W_5W_2$. This can be done first by inserting W_3 between W_1 and W_2, followed by inserting W_4 between W_3 and W_2, followed by inserting W_5 between W_4 and W_2. The other sequence would be first by inserting W_5 between W_1 and W_2, followed by inserting W_4 between W_1 and W_5, followed by inserting W_3 between W_1 and W_4. This shows that ambiguity in gene sequences is also possible (i.e., starting from one sequence we are able to get another sequence in more than one way such that the intermediate sequences are different). This motivates us to study about the ambiguity in gene sequences by using Matrix insertion-deletion systems. Study of this concept of ambiguity may be useful in considering inheritance properties and phylogenetic trees [14]. More specifically, when these intermediate sequences are represented as phylogenetic trees, we can see that the trees are different and thus it might help us to identify the inheritance properties.

With the above mentioned details, in this paper we first introduce a simple and powerful bio-inspired distributed computing model named Matrix insertion-deletion systems. This model is obtained by combining insertion-deletion and matrix grammars. Next, in Section 4 we show that the newly introduced variant can capture the various important and essential bio-molecular structures described in DNA, RNA, protein like *hairpin*, *stem and loop*, *ideal*, *orthodox*, *dumbbell*, *pseudoknot*, *clover-leaf*, *attenuator* and *non-ideal attenuator*. In Section 5, we analyze the application of ambiguity in gene sequences by using the Matrix insertion-deletion system and discuss about the universality result for the new system.

2 Preliminaries

We assume that the readers are familiar with the notions of formal language theory. However, we recall the basic notions which are used in the paper. A finite non-empty set V or Σ is called an alphabet. We denote by V^* or Σ^*, the free monoid generated by V or Σ, by λ it identity or the empty string, and by V^+ or Σ^+ the set $V^* - \{\lambda\}$ or $\Sigma^* - \{\lambda\}$. The elements of V^* or Σ^* are called *words* or *strings*. For any word $w \in V^*$ or Σ^*, we denote the length of w by $|w|$. For more details on formal language theory, we refer to [10]. RE represents the family of recursive enumerable languages.

Next, we will look into the basic definitions of insertion-deletion systems. Given an insertion-deletion system $\gamma = (V, T, A, R)$, where V is an alphabet, $T \subseteq V$, A is a finite language over V, R is a finite triples of the form $(u, \beta/\alpha, v)$, where $(u, v) \in V^*$, $(\alpha, \beta) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$. The pair (u, v) are called as contexts which will be used in insertion/deletion rules. Insertion rule will be of the form $(u, \lambda/\alpha, v)$ which means that α is inserted between u and v. Deletion rule will be of the form $(u, \beta/\lambda, v)$, which means that β is deleted between u and v. In other words, $(u, \lambda/\alpha, v)$ corresponds to the rewriting rule $uv \rightarrow u\alpha v$, and $(u, \beta/\lambda, v)$ corresponds to the rewriting rule $u\beta v \rightarrow uv$.

Consequently, for $x, y \in V^*$ we can write $x \Longrightarrow^* y$, if y can be obtained from x by using either an insertion rule or a deletion rule which is given as follows: (the down arrow $\downarrow$ indicates the position where the string is inserted, the down arrow $\Downarrow$ indicates the position where the string is deleted and the underlined string indicates the string inserted/deleted)

1. $x = x_1 u^{\downarrow} v x_2,\ y = x_1 u \underline{\alpha} v x_2$, for some $x_1, x_2 \in V^*$ and $(u, \lambda/\alpha, v) \in R$.
2. $x = x_1 u \underline{\beta} v x_2,\ y = x_1 u^{\Downarrow} v x_2$, for some $x_1, x_2 \in V^*$ and $(u, \beta/\lambda, v) \in R$.

The language generated by γ is defined by

$$L(\gamma) = \{w \in T^* \mid x \Longrightarrow^* w, for\ some\ x \in A\}$$

where $\Longrightarrow^*$ is the reflexive and transitive closure of the relation $\Longrightarrow$. The family of languages generated by the insertion-deletion systems is given as $INS_n^m DEL_p^q$ for $n, m, p, q \geq 0$, where n denotes the maximal length of the inserted string, m denotes the maximal length of the context in insertion rules, p denotes the maximal length of the deleted string and q denotes the maximal length of the context in deletion rules.

Next, we will look into the definition of matrix grammars. A matrix grammar is an ordered quadruple $G = (N, T, S, M)$ where N is a set of non-terminals, T is a set of terminals, S is the start symbol and M is a finite set of nonempty sequences whose elements are ordered pairs (P, Q). The pairs are referred to as productions and written in the form $P \to Q$. The sequences are referred to as matrices and written $m = [P_1 \to Q_1, ..., P_r \to Q_r], r \geq 1$. Some rules in a matrix are exempted in applying the derivation if those rules are present in the set *appearance checking.* The language generated by the matrix grammar is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages generated by matrix grammars is denoted by MAT^{λ} (the λ on the upper index indicates that $P \to \lambda$ is allowed). For more details on matrix grammars, we refer to [1], [12].

The language of DNA can be considered over $\Sigma_{DNA} = \{a, t, g, c\}$, where the complementary can be given as: $\bar{a} = t$, $\bar{t} = a$, $\bar{g} = c$ and $\bar{c} = g$. Similarly, the language of RNA can be considered over $\Sigma_{RNA} = \{a, u, g, c\}$, where the complementary can be given as: $\bar{a} = u$, $\bar{u} = a$, $\bar{g} = c$ and $\bar{c} = g$.

3 Matrix Insertion-Deletion Systems

In this section, we introduce our new grammar system *Matrix insertion-deletion systems.* A Matrix insertion-deletion system is a construct $\Upsilon = (V, T, A, R)$ where V is an alphabet, $T \subseteq V$, A is a finite language over V, R is a finite triples of the form in matrix format $[(u_1, \beta_1/\alpha_1, v_1), \ldots, (u_n, \beta_n/\alpha_n, v_n)]$, where $(u_k, v_k) \in V^*$, and $(\alpha_k, \beta_k) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$, with $(u_k, \beta_k/\alpha_k, v_k) \in R_{I_i} \cup R_{D_j} \cup R_{I_i/D_j}$, for $1 \leq k \leq n$. Here R_{I_i} denotes the matrix which consists of only insertion rules, R_{D_j} denotes the matrix which consists of only deletion rules and R_{I_i/D_j} denotes the matrix which consists of both insertion and deletion rules.

Consequently, for $x, y \in V^*$ we can write $x \Longrightarrow x' \Longrightarrow x'' \Longrightarrow \ldots \Longrightarrow y$, if y can be obtained from x by using the matrix consisting of insertion or deletion or insertion and deletion rules as follows: (the down arrow $\downarrow$ indicates the position where the string is inserted, the down arrow $\Downarrow$ indicates the position where the string is deleted and the underlined string indicates the string inserted/deleted)

1. $x = x_1u_1 {}^{\downarrow}v_1u_2v_2...u_nv_nx_2 \Longrightarrow x' = x_1u_1\underline{\alpha_1}v_1u_2 {}^{\downarrow}...u_nv_nx_2 \Longrightarrow x'' = x_1u_1\alpha_1v_1u_2\underline{\alpha_2}v_2...u_n {}^{\downarrow}v_nx_2 \Longrightarrow^* y = x_1u_1\alpha_1v_1u_2\alpha_2v_2...u_n\underline{\alpha_n}v_nx_2$, for some $x_1, x_2 \in V^*$ and $[(u_1, \lambda/\alpha_1, v_1), (u_2, \lambda/\alpha_2, v_2), \ldots, (u_n, \lambda/\alpha_n, v_n)] \in R_{I_i}$.
2. $x = x_1u_1\underline{\beta_1}v_1u_2\beta_2v_2...u_n\beta_nv_nx_2 \Longrightarrow x' = x_1u_1 {}^{\Downarrow}v_1u_2\underline{\beta_2}v_2...u_n\beta_nv_nx_2 \Longrightarrow x'' = x_1u_1v_1u_2 {}^{\Downarrow}v_2...u_n\underline{\beta_n}v_nx_2 \Longrightarrow^* y = x_1u_1v_1u_2v_2...u_n {}^{\Downarrow}v_nx_2$ for some $x_1, x_2 \in V^*$ and $[(u_1, \overline{\beta_1/\lambda}, v_1), (u_2, \beta_2/\lambda, v_2), ..., (u_n, \beta_n/\lambda, v_n)] \in R_{D_j}$.
3. $x = x_1u_1 {}^{\downarrow}v_1u_2\beta_2v_2...u_n\beta_nv_nx_2 \Longrightarrow x' = x_1u_1\underline{\alpha_1}v_1u_2\underline{\beta_2}v_2...u_n\beta_nv_nx_2 \Longrightarrow x'' = x_1u_1\alpha_1v_1u_2 {}^{\Downarrow}v_2...u_n\underline{\beta_n}v_nx_2 \Longrightarrow^* y = x_1u_1\alpha_1v_1u_2v_2...u_n {}^{\Downarrow}v_nx_2$, for some $x_1, x_2 \in V^*$ and $[(u_1, \overline{\lambda}/\alpha_1, v_1), (u_2, \beta_2/\lambda, v_2), ..., (u_n, \beta_n/\lambda, v_n)] \in R_{I_i/D_j}$.

In a derivation step the rules in a matrix are applied sequentially one after other in order and no rule is in appearance checking (note that the rules in a matrix are not applied in parallel). The language generated by Υ is defined by

$$L(\Upsilon) = \{w \in T^* \mid x \Longrightarrow^*_{R_\chi} w, \text{ for some } x \in A\}, \text{ where } \chi \in \{I_i, D_j, I_i/D_j\}$$

where $\Longrightarrow^*$ is the reflexive and transitive closure of the relation $\Longrightarrow$. Note that the string w is collected after applying all the rules in a matrix and also $w \in T^*$ only. The family of languages generated by Matrix insertion-deletion systems is given as $MATINS_n^m DEL_p^q$. The following example gives the clear understanding of the Matrix insertion-deletion systems. Consider the triple agreement language $L_1 = \{a^n b^n c^n \mid n \geq 1\}$. The language L_1 can be generated by the following Matrix insertion-deletion system Υ_1.

Example 1. $L_1 = \{a^n b^n c^n \mid n \geq 1\} \in \Upsilon_1$. The language L_1 can be generated by the Matrix insertion-deletion system $\Upsilon_1 = (\{a, b, c\}, \{a, b, c\}, \{abc\}, \{R_{I_1} = [(a, \lambda/ab, b), (b, \lambda/c, c)]\})$. A sample derivation can be given as follows: $a^{\downarrow}bc \Longrightarrow_{R_{I_1}} a\underline{ab}b^{\downarrow}c \Longrightarrow_{R_{I_1}} aabbb\underline{c}c \Longrightarrow^*_{R_{I_1}} a^n b^n c^n$. The triple agreement language has relevances to triple-stranded DNA [6]. If we include d in V, T, replace the axiom as $abcd$ and R_{I_1} as $[(a, \lambda/ab, b), (c, \lambda/cd, d)]$ in Υ_1 we can see that Υ_1 generates quadruple agreement language $\{a^n b^n c^n d^n \mid n \geq 1\}$. The quadruple agreement language has relevances to quadruple-stranded DNA [6].

4 Representing Bio-Molecular Structures

In this section, we show that the Matrix insertion-deletion systems can capture the important and essential biological structures discussed earlier in the paper. In most of the following derivations, at each derivation step, we directly write the resultant string obtained by applying all the rules in a matrix.

Lemma 1. *The pseudoknot structure language $L_{ps} = \{uv\bar{u}^R\bar{v}^R \mid u, v \in \Sigma^*_{DNA}\}$ can be generated by Matrix insertion-deletion system.*

Proof. The language L_{ps} can be generated by the Matrix insertion-deletion system $\Upsilon_{ps} = (\{b, \bar{b}, \dagger_1, \dagger_2, \dagger_3, \dagger_4\}, \{b, \bar{b}\}, \{\lambda, \dagger_1 \dagger_2 \dagger_3 \dagger_4\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as follows:
$R_{I_1} = [(\lambda, \lambda/b, \dagger_1), (\lambda, \lambda/\bar{b}, \dagger_3)]$, $R_{I_2} = [(\lambda, \lambda/b, \dagger_2), (\lambda, \lambda/\bar{b}, \dagger_4)]$
$R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_3/\lambda, \lambda)]$, $R_{D_2} = [(\lambda, \dagger_2/\lambda, \lambda), (\lambda, \dagger_4/\lambda, \lambda)]$
A sample derivation is given as follows:

$$^{\downarrow} \dagger_1 \dagger_2^{\downarrow} \dagger_3 \dagger_4 \Longrightarrow_{R_{I_1}} \underline{a} \dagger_1^{\downarrow} \dagger_2\underline{t} \dagger_3^{\downarrow} \dagger_4 \Longrightarrow_{R_{I_2}} a \dagger_1 \underline{g}^{\downarrow} \dagger_2 t \dagger_3 \underline{c}^{\downarrow}\dagger_4 \Longrightarrow_{R_{I_2}}$$
$$a \dagger_1 g\underline{a} \dagger_2 t \dagger_3 c\underline{t}\dagger_4 \Longrightarrow_{R_{D_1}} a^{\Downarrow}ga \dagger_2 t^{\Downarrow}ct\dagger_4 \Longrightarrow_{R_{D_2}} aga^{\Downarrow}tct^{\Downarrow}$$

The attenuator language in a molecular structure can be represented as $L_{an} = \{u\bar{u}^R u\bar{u}^R \mid u \in \Sigma^*_{DNA}\}$. The Fig.3. shows the attenuator structure.

Lemma 2. *The attenuator language $L_{an} = \{u\bar{u}^R u\bar{u}^R \mid u \in \Sigma^*_{DNA}\}$ can be generated by Matrix insertion-deletion system.*

Proof. The language L_{an} can be generated by the Matrix insertion-deletion system $\Upsilon_{an} = (\{a, t, g, c, \dagger_1, \dagger_2\}, \{a, t, g, c\}, \{\lambda, \dagger_1\dagger_2\}, R)$, where R is given as follows:
$R_{I_1} = [(\lambda, \lambda/a, \dagger_1), (\dagger_1, \lambda/t, \lambda), (\lambda, \lambda/a, \dagger_2), (\dagger_2, \lambda/t, \lambda)]$
$R_{I_2} = [(\lambda, \lambda/t, \dagger_1), (\dagger_1, \lambda/a, \lambda), (\lambda, \lambda/t, \dagger_2), (\dagger_2, \lambda/a, \lambda)]$
$R_{I_3} = [(\lambda, \lambda/c, \dagger_1), (\dagger_1, \lambda/g, \lambda), (\lambda, \lambda/c, \dagger_2), (\dagger_2, \lambda/g, \lambda)]$
$R_{I_4} = [(\lambda, \lambda/g, \dagger_1), (\dagger_1, \lambda/c, \lambda), (\lambda, \lambda/g, \dagger_2), (\dagger_2, \lambda/c, \lambda)]$
$R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_2/\lambda, \lambda)]$

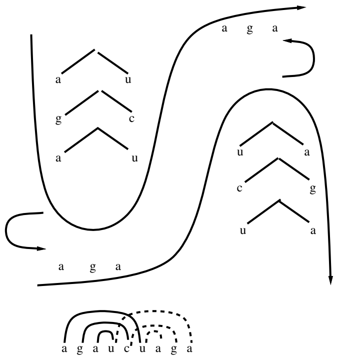


Fig. 3. Attenuator representation

A sample derivation is given as follows:

$${}^{\downarrow}\dagger_1^{\downarrow}\,{}^{\downarrow}\dagger_2^{\downarrow} \Longrightarrow_{R_{I_1}} \underline{a}^{\downarrow}\,\dagger_1^{\downarrow}\,\underline{t}\;\underline{a}^{\downarrow}\,\dagger_2^{\downarrow}\,\underline{t} \Longrightarrow_{R_{I_2}} a\underline{t}^{\downarrow}\,\dagger_1^{\downarrow}\,\underline{a}ta\underline{t}^{\downarrow}\,\dagger_2^{\downarrow}\,\underline{a}t \Longrightarrow_{R_{I_3}} at\underline{c}^{\downarrow}\,\dagger_1^{\downarrow}\,\underline{g}atat\underline{c}^{\downarrow}$$
$$\dagger_2^{\downarrow}\underline{g}at \Longrightarrow_{R_{I_4}} atc\underline{g}\dagger_1\,\underline{c}gatatc\underline{g}\dagger_2\,\underline{c}gat \Longrightarrow_{R_{D_1}} atcg^{\Downarrow}cgatatcg^{\Downarrow}cgat \qquad \square$$

Lemma 3. *The non ideal-attenuator language $L_{nian} = \{u\bar{u}^R u \mid u \in \Sigma_{DNA}^*\}$ can be generated by Matrix insertion-deletion system.*

Proof. The language L_{nian} can be generated by the Matrix insertion-deletion system $\Upsilon_{nian} = (\{a, t, g, c, \dagger_1, \dagger_2\}, \{a, t, g, c\}, \{\lambda, \dagger_1\dagger_2\}, R)$, where R is given as

$$R_{I_1} = [(\lambda, \lambda/a, \dagger_1), (\dagger_1, \lambda/t, \lambda), (\lambda, \lambda/a, \dagger_2)]\; R_{I_2} = [(\lambda, \lambda/t, \dagger_1), (\dagger_1, \lambda/a, \lambda), (\lambda, \lambda/t, \dagger_2)]$$
$$R_{I_3} = [(\lambda, \lambda/c, \dagger_1), (\dagger_1, \lambda/g, \lambda), (\lambda, \lambda/c, \dagger_2)]\; R_{I_4} = [(\lambda, \lambda/g, \dagger_1), (\dagger_1, \lambda/c, \lambda), (\lambda, \lambda/g, \dagger_2)]$$
$$R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_2/\lambda, \lambda)]$$

A sample derivation is given as follows:

$${}^{\downarrow}\dagger_1^{\downarrow}\,{}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_1}} \underline{a}^{\downarrow}\,\dagger_1\,\underline{t}\;\underline{a}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_2}} a\underline{t}^{\downarrow}\,\dagger_1\,\underline{a}ta\underline{t}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_3}} at\underline{c}^{\downarrow}\,\dagger_1\,\underline{g}atat\underline{c}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_4}}$$
$$atc\underline{g}^{\downarrow}\,\dagger_1\,\underline{c}gatatc\underline{g}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_3}} atcg\underline{c}\,\dagger_1\,\underline{g}cgatatcg\underline{c}\dagger_2 \Longrightarrow_{R_{D_1}} atcgc^{\Downarrow}gcgatatcgc^{\Downarrow} \qquad \square$$

Lemma 4. *The hairpin language $L_{hp} = \{w = \bar{w}^R \mid w \in \Sigma_{DNA}^*\}$ can be generated by Matrix insertion-deletion system.*

Proof. The hairpin language $L_{hp} = \{w = \bar{w}^R \mid w \in \Sigma_{DNA}^*\}$ can be generated by the Matrix insertion-deletion system $\Upsilon_{hp} = (\{b, \bar{b}, \dagger\}, \{b\bar{b}\}, \{\lambda, b \dagger \bar{b}\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as follows:

$R_{I_1} = [(\lambda, \lambda/b, \dagger), (\dagger, \lambda/\bar{b}, \lambda)], R_{D_1} = [(\lambda, \dagger/\lambda, \lambda)]$

A sample derivation is given as follows:

$$a^{\downarrow}\,\dagger^{\downarrow}\,t \Longrightarrow_{R_{I_1}} a\underline{t}^{\downarrow}\,\dagger^{\downarrow}\,\underline{a}t \Longrightarrow_{R_{I_1}} at\underline{g}\,\dagger\,\underline{c}at \Longrightarrow_{R_{D_1}} atgcg^{\Downarrow}cgcat \qquad \square$$

Lemma 5. *The stem and loop language $L_{sl} = \{uv\bar{u}^R \mid u, v \in \Sigma^*_{DNA}\}$ can be generated by Matrix insertion-deletion system.*

Proof. The stem and loop language L_{sl} can be generated by the Matrix insertion-deletion system $\Upsilon_{sl} = (\{b, \bar{b}, \dagger_1, \dagger_2, \dagger_3\}, \{b, \bar{b}\}, \{\lambda, b \dagger_1 \dagger_3 \dagger_2 \bar{b}\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as follows:

$R_{I_1} = [(\lambda, \lambda/b, \dagger_1), (\dagger_2, \lambda/\bar{b}, \lambda)]$, $R_{I_2} = [(\lambda, \lambda/b, \dagger_3)]$
$R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_2/\lambda, \lambda)]$, $R_{D_2} = [(\lambda, \dagger_3/\lambda, \lambda)]$

A sample derivation is given follows:

$$a^{\downarrow} \dagger_1 \dagger_3 \dagger_2^{\downarrow}\, t \Longrightarrow_{R_{I_1}} a\underline{c} \dagger_1^{\downarrow} \dagger_3 \dagger_2\, \underline{g}t \Longrightarrow_{R_{I_2}} ac \dagger_1 \underline{t} \dagger_3 \dagger_2 gt \Longrightarrow_{R_{I_2}}$$
$$ac \dagger_1 t\underline{c} \dagger_3 \dagger_2 gt \Longrightarrow_{R_{D_1}} aca^{\Downarrow}tc \dagger_3\, gt \Longrightarrow_{R_{D_2}} acatc^{\Downarrow}gt \qquad \square$$

The dumbbell language in a molecular structure can be represented as $L_{db} = \{u\bar{u}^R v\bar{v}^R \mid u, v \in \Sigma^*_{DNA}\}$. The Fig.4. shows the pictorial representation of dumbbell language.

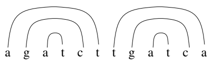


Fig. 4. Dumbbell representation in bio-molecular structures

Lemma 6. *The dumbbell language $L_{db} = \{u\bar{u}^R v\bar{v}^R \mid u, v \in \Sigma^*_{DNA}\}$ can be generated by Matrix insertion-deletion system.*

Proof. The dumbbell language L_{db} can be generated by the Matrix insertion-deletion system $\Upsilon_{db} = (\{b, \bar{b}, \dagger_1, \dagger_2\}, \{b, \bar{b}\}, \{\lambda, \dagger_1\dagger_2\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as follows:

$R_{I_1} = [(\lambda, \lambda/b, \dagger_1), (\dagger_1, \lambda/\bar{b}, \lambda)]$, $R_{I_2} = [(\lambda, \lambda/b, \dagger_2), (\dagger_2, \lambda/\bar{b}, \lambda)]$
$R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda)]$, $R_{D_2} = [(\lambda, \dagger_2/\lambda, \lambda)]$

A sample derivation is given follows:

$${}^{\downarrow} \dagger_1^{\downarrow} \dagger_2 \Longrightarrow_{R_{I_1}} \underline{a} \dagger_1 \underline{t}^{\downarrow} \dagger_2^{\downarrow} \Longrightarrow_{R_{I_2}} a \dagger_1 t\underline{g}^{\downarrow} \dagger_2^{\downarrow}\, \underline{c} \Longrightarrow_{R_{I_2}} a \dagger_1 tg\underline{c} \dagger_2\, \underline{g}c \Longrightarrow_{R_{D_1}}$$
$$a^{\Downarrow}tgc \dagger_2\, gc \Longrightarrow_{R_{D_2}} atgc^{\Downarrow}gc \qquad \square$$

Definition 1. *A string w over a complementary alphabet Σ is called ideal iff $|w|_b = |w|_{\bar{b}}$ for all $b \in \Sigma$. A language is ideal iff it contains only ideal strings.*

Lemma 7. *The ideal language L_{id} can be generated by Matrix insertion-deletion system.*

Proof. The ideal language L_{id} can be generated by the Matrix insertion-deletion system $\Upsilon_{id} = (\{b, \bar{b}\}, \{b, \bar{b}\}, \{\lambda\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as $R_{I_1} = [(\lambda, \lambda/b, \lambda), (\lambda, \lambda/\bar{b}, \lambda)]$.
A sample derivation is given as follows:

$${}^{\downarrow}\lambda^{\downarrow} \Longrightarrow_{R_{I_1}} \underline{a}^{\downarrow}\underline{t}^{\downarrow} \Longrightarrow_{R_{I_1}} a\underline{c}^{\downarrow}t\underline{g}^{\downarrow} \Longrightarrow_{R_{I_1}} ac\underline{t}^{\downarrow}tg\underline{a}^{\downarrow} \Longrightarrow_{R_{I_1}} act\underline{c}tga\underline{g} \qquad \square$$

Definition 2. *A string w over a complementary alphabet Σ is called orthodox iff it is (i) the empty string ϵ, or (2) the result of inserting two adjacent complementary element $b\bar{b}$, for some $b \in \Sigma$, anywhere in an orthodox string. A language is orthodox iff it contains only orthodox strings.*

Lemma 8. *The orthodox language L_{od} can be generated by Matrix insertion-deletion system.*

Proof. The orthodox language L_{od} can be generated by the Matrix insertion-deletion system $\Upsilon_{od} = (\{b, \bar{b}\}, \{b, \bar{b}\}, \{\lambda\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as $R_{I_1} = [(\lambda, \lambda/b\bar{b}, \lambda)]$. A sample derivation is given as $\lambda^{\downarrow} \Longrightarrow_{R_{I_1}} \underline{at}^{\downarrow} \Longrightarrow_{R_{I_1}} a^{\downarrow}t\underline{gc} \Longrightarrow_{R_{I_1}} a\underline{ta}tgc^{\downarrow} \Longrightarrow_{R_{I_1}} atatgc\underline{cg}$ □

The cloverleaf language in bio-molecular structures is represented as $L_{cl} = \{uv_1\bar{v}_1^R v_2\bar{v}_2^R, ..., v_n\bar{v}_n^R\bar{u}^R \mid u, v_1, v_2, v_3, ..., v_n \in \Sigma_{DNA}^*, n \geq 0\}$. The following Fig.5. represents the diagramatic representation of cloverleaf language for $n = 3$.

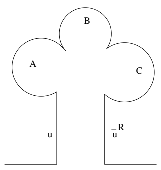


Fig. 5. Cloverleaf representation(where $A = v_1\bar{v}_1^R$, $B = v_2\bar{v}_2^R$, $C = v_3\bar{v}_3^R$)

Lemma 9. *The cloverleaf language $L_{cl} = \{uv_1\bar{v}_1^R v_2\bar{v}_2^R, ..., v_n\bar{v}_n^R\bar{u}^R \mid u, v_1, v_2, ..., v_n \in \Sigma_{DNA}^*, n \geq 0\}$ can be generated by Matrix insertion-deletion systems.*

Proof. The cloverleaf language L_{cl} (for $n = 3$) can be generated by the Matrix insertion-deletion system $\Upsilon_{cl} = (\{b, \bar{b}, \dagger_1, \dagger_2, \dagger_3, \dagger_4, \dagger_4, \dagger_5\}, \{b, \bar{b}\}, \{\lambda, b \dagger_1 \dagger_2 \bar{b}, \dagger_3 \dagger_4 \dagger_5, b \dagger_1 \dagger_3 \dagger_4 \dagger_5 \dagger_2 \bar{b}\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is
$R_{I_1} = [(\lambda, \lambda/b, \dagger_1), (\dagger_2, \lambda/\bar{b}, \lambda)]$, $R_{I_2} = [(\lambda, \lambda/b, \dagger_3), (\dagger_3, \lambda/\bar{b}, \lambda)]$
$R_{I_3} = [(\lambda, \lambda/b, \dagger_4), (\dagger_4, \lambda/\bar{b}, \lambda)]$, $R_{I_4} = [(\lambda, \lambda/b, \dagger_5), (\dagger_5, \lambda/\bar{b}, \lambda)]$
$R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_2/\lambda, \lambda)]$, $R_{D_2} = [(\lambda, \dagger_3/\lambda, \lambda)]$
$R_{D_3} = [(\lambda, \dagger_4/\lambda, \lambda)]$ $R_{D_4} = [(\lambda, \dagger_5/\lambda, \lambda)]$
A sample derivation is given as follows:

$$a^{\downarrow} \dagger_1 \dagger_3 \dagger_4 \dagger_5 \dagger_2^{\downarrow} t \Longrightarrow_{R_{I_1}} a\underline{c}^{\downarrow} \dagger_1 \dagger_3 \dagger_4 \dagger_5 \dagger_2^{\downarrow} \underline{g}t \Longrightarrow_{R_{I_1}} ac\underline{g} \dagger_1^{\downarrow} \dagger_3^{\downarrow} \dagger_4 \dagger_5 \dagger_2 \underline{c}gt$$
$$\Longrightarrow_{R_{I_2}} acg \dagger_1 \underline{t} \dagger_3 \underline{a}^{\downarrow} \dagger_4^{\downarrow} \dagger_5 \dagger_2 cgt \Longrightarrow_{R_{I_3}} acg \dagger_1 \underline{t} \dagger_3 \underline{a}\ \underline{c} \dagger_4 \underline{g}^{\downarrow} \dagger_5^{\downarrow} \dagger_2 cgt \Longrightarrow_{R_{I_4}}$$
$$acg \dagger_1 t \dagger_3 ac \dagger_4 g\underline{a} \dagger_5 \underline{t} \dagger_2 cgt \Longrightarrow_{R_{D_1}} acg^{\Downarrow}t \dagger_3 ac \dagger_4 ga \dagger_5 t^{\Downarrow}cgt \Longrightarrow_{R_{D_2}}$$
$$acgt^{\Downarrow}ac \dagger_4 ga \dagger_5 tcgt \Longrightarrow_{R_{D_3}} acgtac^{\Downarrow}ga \dagger_5 tcgt \Longrightarrow_{R_{D_4}} acgtacga^{\Downarrow}tcgt$$ □

5 Ambiguity Issues in Gene Sequences

In this section, we study the application of various ambiguity levels of insertion-deletion systems introduced in [11]. Since the Matrix insertion-deletion systems is an extension of insertion-deletion systems, the ambiguity levels defined for insertion-deletion system even holds for Matrix insertion-deletion systems also.

Level 0: The Matrix insertion-deletion systems is said to be 0-ambiguous if the same string can be derived from two different axioms. Consider the gene sequence *actgagct* in ideal language. This sequence can be generated by the Matrix insertion-deletion system Υ_{id} from two different axioms *at* and *ta* such that the same string is obtained at the end of the derivation. The two different derivations which differ by axioms are given as follows:

$$Derivation\ 1 : at^{\downarrow} \Longrightarrow a^{\downarrow}t^{\downarrow}\underline{gc} \Longrightarrow a\underline{c}t\underline{g}^{\downarrow}gc^{\downarrow} \Longrightarrow actg\underline{a}gc\underline{t}$$

$$Derivation\ 2 : ta^{\downarrow} \Longrightarrow^{\downarrow} ta^{\downarrow}\underline{gc} \Longrightarrow^{\downarrow} \underline{c}ta\underline{g}gc^{\downarrow} \Longrightarrow \underline{a}ctaggc\underline{t}$$

Level 1: The Matrix insertion-deletion systems is said to be 1-ambiguous if there are two different derivations for the same string which differs by the order of string inserted/deleted. Consider the gene sequence *aagtt* in stem and loop language. This sequence can be generated in two ways by the Matrix insertion-deletion system Υ_{sl}. Note that the axiom for both derivation is same. The two derivations are given as follows:

$Derivation\ 1 : a^{\downarrow} \dagger_1 \dagger_3 \dagger_2^{\downarrow} t \Longrightarrow_{R_{I_1}} a\underline{a} \dagger_1^{\downarrow} \dagger_3 \dagger_2 \underline{t}t \Longrightarrow_{R_{I_2}} aa \dagger_1 \underline{g} \dagger_3 \dagger_2 tt \Longrightarrow_{R_{D_1}}$ $aa^{\Downarrow}g \dagger_3^{\Downarrow} tt \Longrightarrow_{R_{D_2}} aag^{\Downarrow}tt$ (order of insertion is a and g)

$Derivation\ 2 : a \dagger_1^{\downarrow} \dagger_3 \dagger_2 t \Longrightarrow_{R_{I_2}} a^{\downarrow} \dagger_1 \underline{g} \dagger_3 \dagger_2^{\downarrow} t \Longrightarrow_{R_{I_1}} a\underline{a} \dagger_1 g \dagger_3 \dagger_2 \underline{t}t \Longrightarrow_{R_{D_1}}$ $aa^{\Downarrow}g \dagger_3^{\Downarrow} tt \Longrightarrow_{R_{D_2}} aag^{\Downarrow}tt$ (order of insertion is g and a)

Level 2: The Matrix insertion-deletion systems is said to be 2-ambiguous if there are two different derivations for the same string which differs by the order of contexts used for insertion/deletion. Consider the gene sequence *ctaatcgg* in cloverleaf language. This sequence can be generated in two ways by the Matrix insertion-deletion system Υ_{cl}. The two derivations are given as follows:

$Derivation\ 1 : c \dagger_1^{\downarrow} \dagger_3^{\downarrow} \dagger_4 \dagger_5 \dagger_2 g \Longrightarrow_{R_{I_2}} c \dagger_1 \underline{t} \dagger_3 \underline{a}^{\downarrow} \dagger_4^{\downarrow} \dagger_5 \dagger_2 g \Longrightarrow_{R_{I_3}} c \dagger_1 t \dagger_3 a\underline{a}$ $\dagger_4 \underline{t}^{\downarrow} \dagger_5^{\downarrow} \dagger_2 g \Longrightarrow_{R_{I_4}} c \dagger_1 t \dagger_3 aa \dagger_4 t\underline{c} \dagger_5 \underline{g} \dagger_2 g \Longrightarrow_{R_{D_1}} c^{\Downarrow}t \dagger_3 aa \dagger_4 tc \dagger_5 g^{\Downarrow}g$ $\Longrightarrow_{R_{D_3}} ct^{\Downarrow}aa \dagger_4 tc \dagger_5 gg \Longrightarrow_{R_{D_3}} ctaa^{\Downarrow}tc \dagger_5 gg \Longrightarrow_{R_{D_4}} ctaatc^{\Downarrow}gg$

$Derivation\ 2 : c \dagger_1 \dagger_3 \dagger_4^{\downarrow} \dagger_5^{\downarrow} \dagger_2 g \Longrightarrow_{R_{I_4}} c \dagger_1 \dagger_3^{\downarrow} \dagger_4^{\downarrow} \underline{c} \dagger_5 \underline{g} \dagger_2 g \Longrightarrow_{R_{I_3}} c \dagger_1^{\downarrow} \dagger_3^{\downarrow}\underline{a} \dagger_4 \underline{t}c$ $\dagger_5 g \dagger_2 g \Longrightarrow_{R_{I_2}} c \dagger_1 \underline{t} \dagger_3 \underline{a}a \dagger_4 tc \dagger_5 g \dagger_2 g \Longrightarrow_{R_{D_1}} c^{\Downarrow}t \dagger_3 aa \dagger_4 tc \dagger_5 g^{\Downarrow}g \Longrightarrow_{R_{D_2}}$ $ct^{\Downarrow}aa \dagger_4 tc \dagger_5 gg \Longrightarrow_{R_{D_3}} ctaa^{\Downarrow}tc \dagger_5 gg \Longrightarrow_{R_{D_4}} ctaatc^{\Downarrow}gg$

Note that the contexts chosen are of different order in each derivation. The Level 2 ambiguity can be pictorially represented as shown in Fig.6. Fig. 6(a) corresponds to derivation 1 and Fig.6(b) corresponds to derivation 2. This picture suggests a way of handling ambiguity issues in gene sequences and how they can be interpreted and what could be the intermediate sequences of genes in its sequence process.

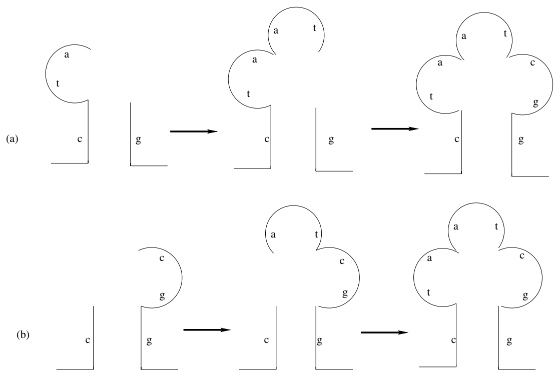


Fig. 6. Ambiguity in cloverleaf language

Level 3: The Matrix insertion-deletion systems is said to be 3-ambiguous if there are two different descriptions for the same string which differs by the position where the string is inserted/deleted. Consider the string *gctagcat* in orthodox language. This string can be derived in two different descriptions by Υ_{od} The two different descriptions are given as follows:

$$Description\ 1 :^{\downarrow} ta \Longrightarrow_{R_{I_1}} \underline{gc}ta^{\downarrow} \Longrightarrow_{R_{I_1}} gcta\underline{gc}^{\downarrow} \Longrightarrow_{R_{I_1}} gctagc\underline{at}$$

$$Description\ 2 : ta^{\downarrow} \Longrightarrow_{R_{I_1}} {}^{\downarrow}ta\underline{gc} \Longrightarrow_{R_{I_1}} \underline{gc}tagc^{\downarrow} \Longrightarrow_{R_{I_1}} gctagc\underline{at}$$

Note that the axiom, order of insertion of strings, order of contexts (here (λ, λ)) all are same in both derivations, but the position of insertion is different in each derivation.

From the above results we can see that there may be more than one way that a gene sequence can be processed.

5.1 Universality of Matrix Insertion-Deletion Systems

Though our aim in this paper is not to analyze the introduced system with respect to Chomsky grammars, we provide the following trivial universality result in order to show that our new system is computationally complete.

Lemma 10. $RE \subseteq MATINS_1^1\ DEL_1^1$

Proof. In [13], it is proved that $RE \subseteq INS_1^1\ DEL_1^1$. Since each rule of insertion-deletion system can be considered as a set of matrices with each matrix consisting a single rule, the above result is true for Matrix insertion-deletion systems also. □

6 Conclusion

Insertion-deletion systems were defined and motivated by the way DNA strands are inserted and deleted. The bio-molecular structures like pseudoknot, attenuator, non-ideal attenuator are beyond the scope of context free grammars.

The bio-molecular structures like hairpin, stem and loop, ideal languages are within the power of context free grammars. We have defined a simple and powerful grammar system named Matrix insertion-deletion system. This is a unique grammar system which encompasses all the important biological structures that can be found in DNA, RNA, protein and other bio molecules. No other grammar system captures all the above discussed bio-molecular structures and therefore this new grammar system deserves a special attention. We have also shown the application of various ambiguity levels of insertion-deletion systems in gene sequences and how the ambiguity can be interpreted in gene sequences. As a future work, it is worth to analyze the closure properties and generative capacity of the introduced Matrix insertion-deletion systems.

References

1. Salomaa, A.: Formal languages. Academic Press, New York (1973)
2. Brendel, V., Busse, H.G.: Genome structure described by formal languages. Nucleic Acids Res., 2561–2568 (1984)
3. Calude, C.S., Paŭn, G.: Computing with cells and atoms, An intro. to Quantum, DNA and Membrane Computing. Taylor and Francis, London (2001)
4. Searls, D.B.: Representing genetic information with formal grammars. In: Proceedings of the National Conference on Artificial Intelligence, pp. 386–391 (1988)
5. Searls, D.B.: The linguistics of DNA. American Scientist, 579–591 (1992)
6. Searls, D.B.: The computational linguistics of biological sequences. In: Hunter, L. (ed.) Artificial Intelligence and Molecular Biology, pp. 47–120. AAAI Press, Menlo Park (1993)
7. Rivas, E., Reddy, S.R.: The language of RNa: A formal grammar that includes psuedoknots. Bioinformatics 16, 334–340 (2000)
8. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing, New Computing Paradigms. Springer, Heidelberg (1998)
9. Păun, G.: Membrane Computing-An introduction. Springer, Heidelberg (2002)
10. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (2006)
11. Krithivasan, K., Kuppusamy, L., Mahendran, A., Khalid, M.: On the ambiguity and complexity measures in insertion-deletion systems. In: Proceedings of Bionetics 2010, USA, December 1-3. LNCS (2010)
12. Rozenberg, G., Salomaa, A.: Handbook of formal languages. Springer, Heidelberg (1997)
13. Verlan, S.: On minimal context-free insertion-deletion systems. Journal of Automata, Languages and Combinatorics 2, 317–328 (2007)
14. Setubal, J.C., Meidanis, J.: Introduction to Computational Molecular Biology. PWS Publishing Company (1997)
15. Uemura, Y., Hasegawa, A., Kobayashi, S., Yokomori, T.: Tree adjoining grammarsfor RNA structure prediction. Theoretical Computer Science 210, 277–303 (1999)

Artificial Bee Colony Based Sensor Deployment Algorithm for Target Coverage Problem in 3-D Terrain

S. Mini[1], Siba K. Udgata[1], and Samrat L. Sabat[2]

[1] Department of Computer and Information Sciences
University of Hyderabad, Hyderabad-500046, India
mini2min2002@yahoo.co.in, udgatacs@uohyd.ernet.in
[2] School of Physics
University of Hyderabad, Hyderabad-500046, India
slssp@uohyd.ernet.in

Abstract. In this paper we address sensor deployment problem to achieve different types of target coverage, viz; simple coverage, k-coverage and Q-coverage. Energy which is an important and scarce resource is not being optimally used if sensor nodes are randomly deployed in a region. This energy wastage can significantly be reduced if the deployment positions can be optimally computed. It is important to provide required coverage by keeping the required sensing range at minimum which will require less energy for sensing. We find out the optimal deployment positions in a 3-D terrain using Artificial Bee Colony (ABC) algorithm, which is based on swarm intelligence, and also compare the sensing range requirement for simple, k and Q-coverage problems. Experimental results reveal that for dense networks, the required sensing range does not increase in same proportion for increased value of k and increased value of average number of sensor nodes in Q for k-Coverage and Q-Coverage problems respectively. Sensitivity analysis is done to study the change in the required sensing range if the sensor nodes cannot be deployed exactly in the optimal positions. The analysis reveals that there is no significant change in the sensing range if the sensor nodes are deployed in near optimal positions.

Keywords: Sensor Deployment, Target Coverage, ABC Algorithm.

1 Introduction

Over the past few years, Wireless Sensor Networks (WSNs) has attracted the attention of researchers. Network lifetime is one crucial element that decides the efficiency of a network. In order to maximize the network lifetime, sensor nodes should be deployed in such a way that energy will be used efficiently. By restricting the sensing range requirement, energy usage can substantially be controlled. When coverage requirement of the targets in a region vary, the problem of deployment becomes complicated. The nodes should be able to provide necessary coverage and the required sensing range should be at minimum.

In general, coverage problems can be classified into two: Area coverage and Target coverage. Area coverage focusses on providing coverage for an entire region, and target coverage aims at providing coverage for certain point objects located in a region. Target coverage can be categorized as simple, k and Q-coverage. A target is required to be monitored by at least one sensor node for simple coverage problem. k-coverage problem arises when all the targets need to be monitored by at least k sensor nodes, where k is a predefined integer constant. Simple coverage problem is a special case of k coverage where $k = 1$. In case of node failures or to increase the accuracy of monitoring, higher values of k are preferred. When all targets $\mathrm{T} = \{T_1, T_2, \ldots, T_n\}$ should be monitored by $\mathrm{Q} = \{q_1, q_2, \ldots, q_n\}$ number of sensor nodes such that target T_j is monitored by at least q_j number of sensor nodes, where n is the number of targets and $1 \leq j \leq n$, it is defined as Q-coverage problem.

Swarms use their environment and resources effectively by collective intelligence. Self-organization is a key feature of a swarm system which gives global solutions/responses through many lower level interactions. Honey bee swarm is an interesting swarm in nature which uses dynamic task allocation and also adapts to environmental changes [1]. We use ABC algorithm, which is currently used to solve many optimization problems, to find sensor node deployment positions such that the coverage requirement is satisfied and required sensing range is optimal.

The rest of the paper is organized as follows: Section 2 presents an overview of related work. In Section 3, the problem is defined. We present the proposed method in Section 4. The proposed method is evaluated by simulations in Section 5. Section 6 concludes the paper.

2 Related Work

Most works on target coverage problem concentrates on finding schedules such that all the nodes need not be active at the same time. Simple coverage [2][3], k-coverage [4][5][6] and Q-coverage [7][8] algorithms for network lifetime maximization focuses on creation of schedules which represents the set of all sensor nodes that should be active for each time instant. Based on this schedule, sensor nodes alternate between active and idle states. Since all sensor nodes need not be active all the time, some energy is preserved and this prolongs the network lifetime. These types of coverage problem assume that all sensor nodes are randomly deployed and the WSN under consideration is a densely deployed network.

Methods to determine the minimum number of sensors to be deployed in a region is presented in [9] and [10]. Clouqueur et al. [9] use the minimum exposure as a measure of the goodness of deployment and aims to maximize the exposure of the least exposed path in the region. Path exposure is a measure of the likelihood of detecting a target traversing the region using a given path. The higher the path exposure, the better the deployment. The set of paths to be considered may be constrained by the environment. They found that the optimal number of sensors deployed in each step varies with the relative cost assigned to deployment and sensors. Watfa et al. [10] consider a 3-D region and finds out the minimum number of nodes required to carry out target coverage. Random deployment, as well as deployment using square and hexagonal lattices is studied for different values of the sensing radius. A measure of optimality was proposed that compares a given deployment of WSN with optimum deployment. This metric is shown to be indicative of the energy efficiency of the WSN and serves as a useful means to select between two different deployments of a WSN. Andersen et al. [11] present an approach called discretization which models sensor deployment problem as a discrete optimization problem. This method does not assure k-coverage of the complete region. In their setting, due to the presence of walls through which a sensor may or may not be able to sense, the region monitored by a sensor is usually not a sphere, but could be of a more complex shape.

Bee colony based algorithms are surveyed by Karaboga et al. [1]. A comparison of ABC with traditional back propagation algorithm and the genetic algorithm done by Karaboga et al. [12] shows that ABC outperforms the other two and can be used to train feed forward neural networks. Another comparison of ABC with Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms, by Karaboga et al. [13], shows that ABC performs better and can be used to solve multimodal engineering problems with high dimensionality. ABC based method to solve area coverage problem for a two dimensional irregular space is proposed by Udgata et al. [14]. The area under consideration is a two dimensional irregular terrain and the objective is to find the sensor node deployment positions in order to minimize the sensing range for simple coverage problem.

In this paper, we deal with sensor deployment problem to address all types of target coverage requirements.

3 Problem Definition

3.1 Sensor Coverage

A sensor node located at (x_1, y_1, z_1) can cover a target at (x_2, y_2, z_2) if the euclidean distance between the sensor node and the target is less than or equal to the sensing range sr.

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \leq sr \tag{1}$$

3.2 Mean of Location Points

The mean value of the location points (x_r, y_r, z_r) for $r = 1, 2, \ldots, N$, is represented by (a_1, a_2, a_3), where

$$a_1 = \frac{\sum_{r=1}^{N}(x_r)}{N} \tag{2}$$

$$a_2 = \frac{\sum_{r=1}^{N}(y_r)}{N} \tag{3}$$

$$a_3 = \frac{\sum_{r=1}^{N}(z_r)}{N} \tag{4}$$

3.3 Simple Coverage

Given a set of targets T = $\{T_1, T_2, \ldots, T_n\}$ located in $u \times v \times w$ region and a set of sensor nodes S = $\{S_1, S_2, \ldots, S_m\}$, the objective is to deploy the sensor nodes such that all the targets are continuously monitored and the network lifetime is extended by keeping the sensing range at minimum. In other words, the objective is to cover all the targets in a given region by at least one sensor node and to minimize the function

$$F = \forall_i((max(distance(S_i, P_g)))) \tag{5}$$

where P is the set of all targets monitored by S_i, $i = 1, 2, \ldots, m$, $g = 1, 2, \ldots, h$, where h is the total number of targets that the sensor node S_i monitors.

3.4 k-Coverage

Given a set of targets T = $\{T_1, T_2, \ldots, T_n\}$ located in $u \times v \times w$ region and a set of sensor nodes S = $\{S_1, S_2, \ldots, S_m\}$, the objective is to deploy the sensor nodes such that all T = $\{T_1, T_2, \ldots, T_n\}$ is covered by at least k number of sensor nodes, $1 \leq k \leq m$ and the network lifetime is extended by keeping the sensing range at minimum. In other words, the objective is to cover all the targets in a given region by at least k sensor nodes and to minimize F denoted by eqn. 5.

3.5 Q-Coverage

Given a set of targets T = $\{T_1, T_2, \ldots, T_n\}$ located in $u \times v \times w$ region and a set of sensor nodes S = $\{S_1, S_2, \ldots, S_m\}$, the objective is to deploy the sensor nodes such that all T = $\{T_1, T_2, \ldots, T_n\}$ is covered by at least Q = $\{q_1, q_2, \ldots, q_n\}$ sensor nodes, where each target T_j, $1 \leq j \leq n$, is covered by at least q_j sensor nodes, $1 \leq q_j \leq m$, at any time and the network lifetime is extended by keeping the sensing range at minimum. In other words, the objective is to cover each target by at least q_j sensor nodes and to minimize F denoted by eqn. 5.

3.6 Cluster Formation

Partitioning the targets into clusters will be a key to identify the position of sensor nodes. Each sensor node is associated to a cluster. Let the set of clusters to be formed be represented as $C = \{C_1, C_2, \ldots, C_m\}$. A target T_j belongs to C_i if and only if $distance(T_j, S_i) \leq distance(T_j, S_l)\forall_l$ where $l = 1, 2, \ldots, m$; $l \neq i$ and $j = 1, 2, \ldots, n$. After computing clusters, if any $C_i = \phi$, mark $C = C - \{C_i\}$ it implies that S_i is not associated to any cluster. For simple coverage problem, each target is associated to exactly one cluster and for k and Q-coverage problems, each target is associated to minimum of k and q_j sensor nodes respectively.

```
1: for each $B_e$ do
2:     $var = 0$
3:     repeat
4:         if $var = 0$ then
5:             Calculate distance between each target and all the sensor locations
6:             Form clusters by assigning targets to $1/k/Q$ sensor nodes which
               are at minimum distance (Sec. 3.4)
7:             if all sensor nodes form cluster then
8:                 Move the sensor location to centroid of all target location points
                   that are associated with it
9:                 $var = 1$
10:            else
11:                Move sensors without assigned targets to random target loca-
                   tions
12:            end if
13:        end if
14:    until $var = 1$
15: end for
```

Fig. 1. Pseudocode: Cluster Formation

4 Proposed Approach

4.1 Behavior of Bees in Nature

Honey bee swarms consists of three essential components: food sources, employed foragers and unemployed foragers. The model defines two leading modes of the behaviour: the recruitment to a nectar source and the abandonment of a source. Initially, a potential forager will start as unemployed forager. That bee will have no knowledge about the food sources around the nest. This bee can be a scout and starts searching around the nest spontaneously for a food due to some internal motivation or possible external clue. After locating the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. Hence, the bee will become an employed forager.

The foraging bee takes a load of nectar from the source and returns to the hive and unloads the nectar to a food store. After unloading the food, the bee performs a special form of dance called waggle dance[15] which contains information about the direction in which the food will be found, its distance from the hive and its quality rating. Since information about all the current rich sources is available to an onlooker on the dance floor, an onlooker bee probably could watch numerous dances and choose to employ itself at the most qualitative source. There is a greater probability of onlookers choosing more qualitative sources since more information is circulating about the more qualitative sources. Employed foragers share their information with a probability, which is proportional to the quality of the food source. Hence, the recruitment is proportional to quality of a food source[13][16].

```
1: Initialize the solution population $B$
2: Evaluate fitness
3: Produce new solutions based on cluster centroids
4: Choose the fittest bee
5: $cycle = 1$
6: repeat
7:    Search for new solutions in the neighborhood
8:    if new solution better than old solution then
9:       Memorize new solution and discard old solution
10:   end if
11:   Replace the discarded solution with a newly randomly generated solu-
      tion through a scout bee
12:   Memorize the best solution
13:   $cycle = cycle + 1$
14: until cycle = maximumcycles
```

Fig. 2. Pseudocode: Proposed Method

4.2 Proposed Method

The target locations are assumed to be stationary. A solution is a set of locations where the sensor nodes can be deployed to cover all the targets and sensing range is optimal. Initial solutions are randomly generated. Let the solution population be B. Each solution corresponding to a bee e is denoted as $B_e = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_m, y_m, z_m)\}$ where $e = 1, 2, \ldots, d$, d represents total number of bees and m represents total number of nodes to be deployed.

The initial task is to form clusters according to their location. Each cluster has a sensor node associated as cluster centroid with it. The Euclidean distances of the targets and the sensor locations are calculated. Clusters are formed based on this distance measure. Clusters are generated in such a way that no sensor location in a solution is left idle without being part of a cluster. The number of targets in a cluster will be less if sensor to which the cluster is associated is located at a remote place. The number of clusters formed is exactly equal to the number of sensor nodes to be deployed. The employed bees return with the solution having cluster centroids. All the deployment locations in a solution is replaced by the corresponding cluster centroid. The pseudocode for forming clusters is given in Fig. 1.

The Euclidean distance between each target and the sensor location to which it is associated is used as the fitness function to evaluate the solutions. Let $D_i = (D_{i1}, D_{i2}, D_{i3})$ be the cluster centroid of i^{th} cluster. $F(D_i)$ refers to the nectar amount at food source located at D_i. After watching the waggle dance of employed bees, an onlooker goes to the region of D_i with probability p_i defined as,

$$p_i = \frac{F(D_i)}{\sum_{f=1}^{nf} F(D_f)} \qquad (6)$$

where nf is the total number of food sources. The onlooker finds a neighborhood food source in the vicinity of D_i by using,

$$D_i(t+1) = D_i(t) + \delta_{id} \times v \tag{7}$$

where δ_{id} is the neighborhood patch size for d^{th} food source, v is random uniform variate $\in$ [-1, 1] and t is the cycle number. The onlooker bee then evaluates the fitness function based on the new value $D_i(t+1)$. It should be noted that the solutions are not allowed to move beyond the edge of the region. The new solutions are also evaluated and compared using the fitness function. If any new solution is better than the existing one, the new one is retained and old one is discarded. Scout bees search for a random feasible solution. The solution with the least sensing range is finally chosen as the best solution. The pseudocode of proposed method is given in Fig. 2.

Table 1. Sensing Range Requirement for k-coverage problem

			k=1				k=3				k=5			
N.T [1]	Instance	N.S [2]	Best	Mean	S.D [3]	S.A [4]	Best	Mean	S.D[3]	S.A[4]	Best	Mean	S.D.[3]	S.A [4]
		10	38.88	38.88	0	38.91	92.23	93.47	1.07	92.4	109.89	109.99	0.16	109.96
	1	20	24.34	25.37	0.92	24.41	53.61	53.94	0.54	53.68	64.96	64.96	0	65.08
		30	19.45	20	0.96	19.59	41.76	41.82	0.05	41.85	56.08	56.17	0.15	56.16
		10	39.38	40.07	0.83	39.42	90.91	91.25	0.29	91	104.15	104.7	0.47	104.21
100	2	20	24.61	25.34	1.05	24.72	54.43	54.86	0.61	54.51	61.62	61.62	0	61.7
		30	19.46	19.72	0.22	19.53	46.91	47.6	0.59	46.99	57.17	58.87	1.54	57.2
		10	35.66	36.55	0.98	35.78	87.77	88.38	0.61	87.84	108.71	108.9	0.32	108.86
	3	20	26.2	26.46	0.46	26.23	55.24	55.52	0.24	55.32	65.37	65.79	0.74	65.45
		30	18.79	19.18	0.6	18.91	39.51	39.51	0	39.59	59.43	59.44	0.01	59.53
		10	38.21	38.21	0	38.35	86.31	86.7	0.34	86.44	106.06	106.55	0.74	106.16
	1	20	26.53	27.22	0.6	26.93	53.63	54.38	0.65	53.73	67.34	67.81	0.78	67.5
		30	21.48	21.8	0.45	21.56	41.99	42.24	0.42	42.32	56.41	56.68	0.45	56.51
		10	39.53	39.8	0.48	39.61	99.44	99.89	0.59	99.91	108.68	108.68	0	108.74
150	2	20	26.64	26.89	0.22	26.7	54.62	55.54	1.33	54.83	70.06	70.22	0.14	70.12
		30	21.02	21.39	0.33	21.12	43.23	43.67	0.43	43.51	62.05	62.05	0	62.14
		10	40.93	40.93	0	41	91.05	91.09	0.08	91.13	109.16	109.78	1.08	109.28
	3	20	25.7	26.63	0.8	25.85	53.86	54.34	0.42	53.96	66.21	66.21	0	66.3
		30	21.39	22.28	0.91	21.5	42.81	43.37	0.65	42.92	58.6	58.83	0.37	58.73
		10	42.24	42.3	0.08	42.33	95.85	96.58	0.68	95.96	111.42	111.85	0.69	111.54
	1	20	30.14	30.29	0.27	30.22	58.95	59.71	0.71	59.05	70.6	70.6	0	70.72
		30	23.39	23.71	0.56	23.52	47.74	48.37	0.85	47.9	66	66.96	0.92	66.11
		10	41.22	42.03	1.13	41.32	99.37	99.77	0.35	99.49	111.54	111.74	0.17	111.63
200	2	20	28.95	29.09	0.17	29.08	56.74	56.94	0.34	56.89	70.22	70.22	0	70.34
		30	23.18	23.43	0.24	23.36	47.13	47.13	0	47.56	59.36	59.8	0.38	59.45
		10	42.51	43.08	0.64	42.68	98.06	98.12	0.06	98.19	114.09	114.27	0.27	114.18
	3	20	29.57	29.73	0.28	29.66	59.2	59.72	0.53	59.34	71.54	71.88	0.6	71.6
		30	24.01	24.54	0.77	24.13	47.49	47.73	0.25	47.61	62.02	63.17	1.53	62.09
		10	41.27	41.75	0.67	41.33	99.53	99.85	0.3	99.6	110.04	110.29	0.22	110.1
	1	20	29.68	30.14	0.46	22.73	56.7	57.08	0.33	56.8	67.94	67.97	0.03	68
		30	23.83	24.78	0.83	23.89	44.92	45.89	1.11	44.98	64.59	65.22	0.72	64.63
		10	41.76	41.93	0.29	41.83	96.86	96.99	0.12	96.94	108.48	108.48	0	108.53
250	2	20	28.91	29.59	0.76	28.96	55.24	55.14	0.91	55.27	72.95	73.39	0.38	72.98
		30	23.02	23.87	0.99	23.07	45.3	45.74	0.59	45.38	60.79	60.96	0.16	60.84
		10	42.6	42.72	0.12	42.64	98.26	99.85	1.39	98.31	108.23	108.23	0	108.3
	3	20	28.37	29.39	0.88	28.44	59.58	59.92	0.5	59.64	71.82	72.28	0.45	71.89
		30	23.64	25.19	1.36	23.7	43.78	43.95	0.31	43.84	63.21	63.53	0.44	63.27

[1] Number of targets.
[2] Number of sensor nodes.
[3] Standard Deviation.
[4] Sensitivity Analysis.

5 Results and Discussion

We consider a 200 × 200 × 20m region for experiments. The number of bees is taken as 10, number of cycles is 500 and the number of runs is 3. We conducted experiments using MATLAB 7.

Table 2. Sensing Range Requirement for Q-coverage problem

			Q=1-5				Q=3-5			
N.T[1]	Instance	N.S[2]	Best	Mean	S.D[3]	S.A[4]	Best	Mean	S.D[3]	S.A[4]
100	1	10	111.64	112.29	0.65	111.75	125.27	126.16	1.15	125.34
		20	65.03	65.96	0.81	66.03	89.5	89.65	0.17	89.65
		30	48.25	48.49	0.29	48.39	61.2	62.24	1.12	61.29
	2	10	112.96	115.28	2.01	113.1	146.83	147.63	1.01	146.96
		20	65.81	66.71	1.33	65.93	96.11	97.48	1.27	96.24
		30	48.48	49.14	0.58	48.62	65.11	65.79	0.62	65.2
	3	10	98.5	99.14	0.58	98.64	134.47	135.21	0.93	134.52
		20	61.17	61.62	0.4	61.25	93.5	93.93	0.62	93.58
		30	47.65	48.33	0.8	47.74	65.16	65.77	0.53	65.29
150	1	10	111.57	112.89	1.24	111.64	132.4	132.97	0.54	132.53
		20	70.76	71.16	0.41	70.89	93.12	93.72	0.65	93.22
		30	56.06	56.6	0.49	56.1	70.19	70.48	0.41	70.28
	2	10	109.87	110.16	0.29	109.98	133.55	133.93	0.43	133.67
		20	68.19	68.51	0.53	68.3	99.69	99.97	0.33	99.88
		30	55.83	56.04	0.23	55.89	72.07	72.12	0.06	72.19
	3	10	119.52	120.06	0.58	119.66	143.1	144.05	0.95	143.14
		20	72.36	72.78	0.49	72.5	99.59	101.81	1.96	99.74
		30	59.92	60.32	0.32	59.99	68.53	69.1	0.61	68.6
200	1	10	122.21	122.74	0.51	122.27	138.19	139.06	0.78	138.24
		20	78.75	79.52	0.73	78.81	101.28	101.88	0.55	101.31
		30	63.42	63.5	0.07	63.5	77.6	77.98	0.41	77.69
	2	10	123.74	124.6	0.74	123.8	142.2	143.88	1.61	142.25
		20	71.36	72.17	0.8	71.43	104.95	106.23	1.11	105
		30	59.83	60.67	0.74	59.88	68.49	68.89	0.35	68.54
	3	10	117.21	117.24	0.06	117.28	140.37	142.09	1.6	140.44
		20	70.14	70.22	0.11	70.21	101.79	101.95	0.25	101.83
		30	54.04	55	0.84	54.09	75.82	76.04	0.78	75.89
250	1	10	116.67	117.16	0.49	116.74	130.84	131.56	0.66	130.88
		20	74.47	74.86	0.56	74.54	97.89	98.14	0.41	97.91
		30	58.73	59.85	1	58.8	72.77	73.11	0.51	72.83
	2	10	125.74	127.41	1.46	125.79	139.19	139.33	0.13	139.24
		20	77.38	78.85	1.37	77.42	103.91	105.83	1.82	103.96
		30	57.3	57.94	0.56	57.36	72.83	73.21	0.33	72.9
	3	10	119.69	122.53	2.73	119.74	151.06	152.95	1.68	151.09
		20	77.12	78	0.94	77.17	117.98	119.39	1.44	118.01
		30	57.67	58.1	0.51	57.7	79.47	81.41	1.77	79.53

[1] Number of targets.
[2] Number of sensor nodes.
[3] Standard Deviation.
[4] Sensitivity Analysis.

5.1 Impact of Varying *k* and Q

The value of k is initially set as 1, which implies simple coverage problem. The optimal deployment locations and the required sensing range are computed using the proposed method. The same is done for $k = 3$ and $k = 5$. An increase in sensing range is observed but it is evident that the sensing range requirement does not increase in proportion to the increase in k. The same is observed for Q-coverage requirement also. Q which had values ranging from 1 to 5 and 3 to 7 were used as coverage requirement criteria. Table 1 and Table 2 show the sensing range requirement for k and Q coverage problems respectively. Fig. 3. shows an instance where 10 sensor nodes has to be deployed in a region and k

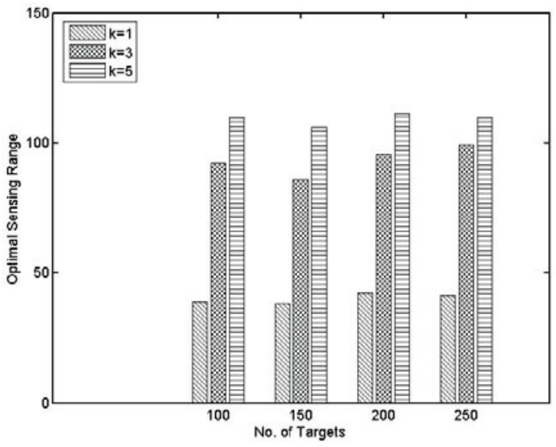


Fig. 3. Sensing range requirement for k-coverage problem where number of sensor nodes to be deployed is 10

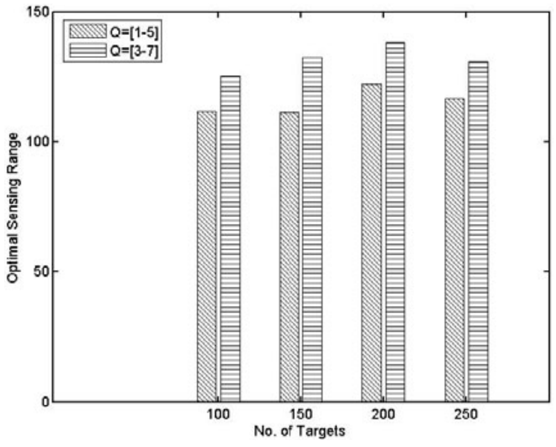


Fig. 4. Sensing range requirement for Q-coverage problem where number of sensor nodes to be deployed is 10

takes values 1, 3 and 5. Fig. 4. shows an instance where Q-coverage requirement has to be satisfied with the least required sensing range. Both the figures clearly show that sensing range requirement does not increase in proportion with the coverage requirement.

5.2 Impact of Varying Number of Sensor Nodes

The number of sensor nodes to be deployed in the region is varied from 10 to 30. The number of clusters increases as the number of sensor nodes increase. The sensing range requirement decreases when more number of nodes are to be deployed. Fig. 5. and Fig. 6. show this decrease in sensing range requirement

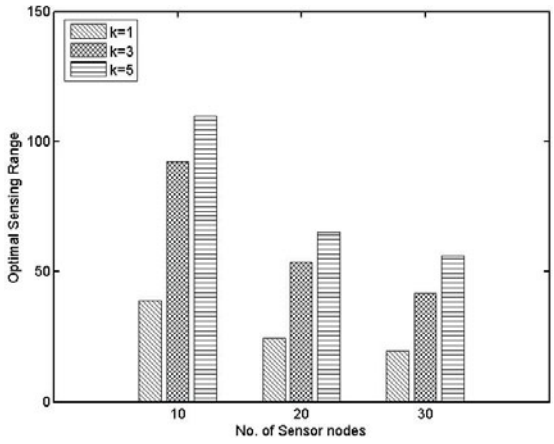


Fig. 5. Sensing range requirement for *k*-coverage problem where number of targets is 100

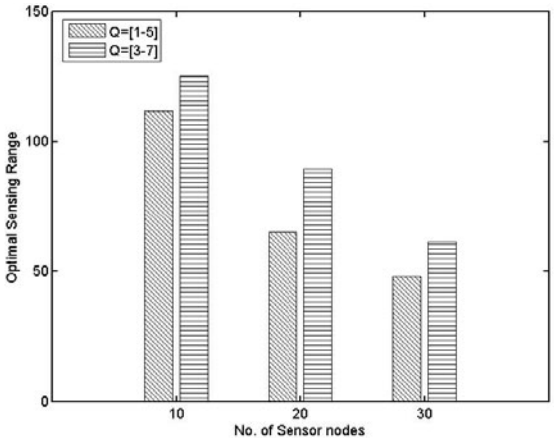


Fig. 6. Sensing range requirement for Q-coverage problem where number of targets is 100

when the number of sensor nodes are increased, for *k* and Q coverage problems respectively.

5.3 Impact of Varying Number of Targets

The number of targets to be covered is varied from 100 to 250. Results show that the sensing range requirement need not essentially be high for higher number of targets. Sensing range requirement is highly dependent on the location of the targets to be covered. The results can also be used to find the minimum number of sensor nodes required to cover specific number of targets with a given sensing range in the 3-D region.

5.4 Sensitivity Analysis

Since it may be hard to deploy the sensors exactly at positions where sensing range is optimal, we conduct sensitivity analysis. We have changed the optimum deployment positions by ± 0.05 and calculated the new required sensing range. The variation in required sensing range is found to be of less significance. The analysis reveals that the deployment solutions obtained through the proposed ABC based method is a robust one and does not change significantly with a slight variation in the optimal deployment positions.

6 Conclusion

In this paper, we have proposed an ABC based method to find optimum sensor deployment positions in a 3-D terrain in order to satisfy different target coverage criteria, namely, simple, k-coverage and Q-coverage. Extensive simulations are carried out with varying number of sensor nodes, number of targets, k-values and values of vector Q to find the minimum sensing range requirement. We notice that sensing range requirement does not increase in same proportion with increase in k or Q requirements. An increase in number of sensor nodes to be deployed, decreases the sensing range requirement. But for a given number of sensor nodes, an increase in the number of targets to be covered need not always make the sensing range requirement high. This method is also suitable to find the optimal number of sensor nodes required to satisfy a coverage criteria. We propose to compare this proposed method with other swarm intelligence techniques in future.

References

1. Karaboga, D., Akay, B.: A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review 31, 61–85 (2009)
2. Slijepcevic, S., Potkonjak, M.: Power Efficient Organization of Wireless Sensor Networks. In: IEEE International Conference on Communications, pp. 472–476 (2001)
3. Cardei, M., Du, D.: Improving Wireless Sensor Network Lifetime through Power Aware Organization. ACM Wireless Networks 11, 333–340 (2005)
4. Huang, C., Tseng, Y.: The Coverage Problem in a Wireless Sensor Network. In: 2nd ACM International Conference on Wireless Sensor Networks and Applications, pp. 115–121 (2003)
5. Yen, L., Yu, C., Cheng, Y.: Expected k-Coverage in Wireless Sensor Networks. Ad Hoc Networks 4, 636–650 (2006)
6. Hefeeda, M., Bagheri, M.: Randomized k-Coverage Algorithms for Dense Sensor Networks. In: INFOCOM, pp. 2376–2380 (2007)
7. Gu, Y., Liu, H., Zhao, B.: Target Coverage with QoS Requirements in Wireless Sensor Networks. In: IPC, pp. 35–38 (2007)
8. Chaudhary, M., Pujari, A.K.: Q-coverage problem in wireless sensor networks. In: Garg, V., Wattenhofer, R., Kothapalli, K. (eds.) ICDCN 2009. LNCS, vol. 5408, pp. 325–330. Springer, Heidelberg (2008)

9. Clouqueur, T., Phipatanasuphorn, V., Ramanathan, P., Saluja, K.: Sensor Deployment Strategy for Detection of Targets Traversing a Region. Mobile Networks and Applications 8, 453–461 (2003)
10. Watfa, M., Commuri, S.: Optimal 3-Dimensional Sensor Deployment Strategy. In: IEEE CCNC 2006, pp. 892–896 (2006)
11. Andersen, T., Tirthapura, S.: Wireless Sensor Deployment for 3D Coverage with Constraints. In: Proc. of the 6th International Conference on Networked Sensing Systems, pp. 78–81 (2009)
12. Karaboga, D., Akay, B., Ozturk, C.: Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In: Torra, V., Narukawa, Y., Yoshida, Y. (eds.) MDAI 2007. LNCS (LNAI), vol. 4617, pp. 318–329. Springer, Heidelberg (2007)
13. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing 8, 687–697 (2008)
14. Udgata, S.K., Sabat, S.L., Mini, S.: Sensor Deployment in Irregular Terrain using ABC Algorithm. In: IEEE BICA 2009, pp. 296–300 (2009)
15. Riley, J.R., Greggers, U., Smith, A.D., Reynolds, D.R., Menzel, R.: The flight paths of honeybees recruited by the waggle dance. Nature 435, 205–207 (2005)
16. Tereshko, V., Loengarov, A.: Collective decision-making in honey bee foraging dynamics. Comput. Inf. Syst. J. 9, 1–7 (2005)

Author Index

Anghelescu, A. 126

Babu, A. Vinaya 162
Baeten, Jos C.M. 35
Banerjee, Satyajit 150
Bhatnagar, Shalabh 175
Bhuvaneswaran, R.S. 290
Brown, Gary 55

Chen, Tzu-Chun 55

Datta Chowdhury, Atish 150
Deswarte, Yves 15
Devaraj, P. 290

Geetha, V. 138
Ghosh, Subhas Kumar 150
Gore, M.M. 209

Honda, Kohei 55

Kiran, Prabhat 197
Krishna, Shankara Narayanan 1, 301
Kuppusamy, Lakshmanan 301

Lakshmanan, K. 175
Lenin, R.B. 126
Lin, Yunfeng 112
Luttik, Bas 35

Mahendran, Anand 301
Mini, S. 313
Mishra, Manas Kumar 209
Misra, C. 255
Mohanty, Hrushikesha 243
Mukhamedov, Aybek 55

Panda, Brajendra S. 187, 267
Parimala, N. 221
Pratt, Vaughan 92
Pushparaj Shetty, D. 187

Raju, K.V.S.V.N. 278
Ramana, K.V. 278
Ramaswamy, S. 126
Rao, S.V. 197
Rekha, J. Ujwala 162
Roşoiu, Ionuţ 232

Sabat, Samrat L. 313
Saini, Anu 221
Sam, I. Shatheesh 290
Shahu Chatrapati, K. 162
Shen, Zhiming 112
Sinha, Koushik 150
Sreenath, N. 138
Sri Krishna, A. 278
Swain, P.K. 255

Udgata, Siba K. 313

Valli Kumari, V. 278
van Tilburg, Paul 35
Varma, N. Sandeep 278

Wimmer, Maria A. 76
Wu, Ryan 112
Wu, Xiaoxin 112

Yaseen, Qussai 267
Yoshida, Nobuko 55
Yoshigoe, K. 126